Revisiting Discrete Logarithm Reductions

Maiara F. Bollauf[1], Roberto Parisella[2] and Janno Siim[1,2]

[1] University of Tartu, Tartu, Estonia
[2] Simula UiB, Bergen, Norway

Abstract. A reduction showing that the hardness of the discrete logarithm (DL) assumption implies the hardness of the computational Diffie-Hellman (CDH) assumption in groups of order p, where $p-1$ is smooth, was first presented by den Boer [Crypto, 88]. We also consider groups of prime order p, where $p-1$ is somewhat smooth (say, every prime q that divides $p-1$ is less than 2^{100}). Several practically relevant groups satisfy this condition.

1. We present a concretely efficient version of the reduction for such groups. In particular, among practically relevant groups, we obtain the most efficient and tightest reduction in the literature for BLS12-381, showing that DL = CDH.
2. By generalizing the reduction, we show that in these groups the n-Power DL (n-PDL) assumption implies n-Diffie-Hellman Exponent (n-DHE) assumption, where n is polynomial in the security parameter.

On the negative side, we show there is no generic reduction, which could demonstrate that n-PDL implies the n-Generalized Diffie-Hellman Exponent (n-GDHE) assumption. This is in stark contrast with the algebraic group model, where this implication holds.

Keywords: Discrete logarithm · Computational Diffie-Hellman · den Boer's reduction · Diffie-Hellman exponent · Generic group model

1 Introduction

The discrete logarithm (DL) assumption postulates that in certain groups, it is hard to compute x given a group generator g and a random group element g^x. The computational Diffie-Hellman (CDH) assumption states that given g and random group elements g^x, g^y, it is hard to compute g^{xy}.

It is well-known that if the CDH assumption is hard, then so is the DL assumption. If the DL assumption was easy, a CDH adversary could compute the discrete logarithm of g^y and output $(g^x)^y$. The technique to analyze the opposite direction was first developed by den Boer [den90] and later generalized by Maurer [Mau94] (see Appendix A for more details). In particular, Maurer showed that in a group of prime[1] order p, the hardness of the DL assumption implies the hardness of the CDH assumption, assuming the reduction receives as an additional auxiliary input some "algebraic" group F with a smooth order[2] defined over the finite field $\mathbb{F}_p$. We call this the den Boer-Maurer (dBM) reduction. More concretely, Maurer suggests using an elliptic curve $E(\mathbb{F}_p)$ with a smooth order as an auxiliary group, whereas den Boer's original approach used $\mathbb{F}^* = \mathbb{F} \setminus \{0\}$.

The practical implications of these results remained unclear since finding smooth elliptic curves over $\mathbb{F}_p$ is a non-trivial problem. Some smooth auxiliary elliptic curve groups were

E-mail: maiarabollauf@gmail.com (Maiara F. Bollauf), robertoparisella@hotmail.it (Roberto Parisella), jannosiim@gmail.com (Janno Siim)

[1] In fact, [Mau94] even considers composite order groups, but we focus on the prime order for simplicity.

[2] We say that an integer $n > 1$ is α-smooth if all primes p_i dividing n satisfy $p_i \leq 2^\alpha$ for some small value α.

constructed by Muzereau, Smart, and Vercauteren [MSV04] for standard elliptic curve groups of that time. More recently, May and Schneider [MS23] constructed concrete smooth elliptic curves for more modern standard elliptic curve groups (NIST P-256, Curve25519, SM2, etc.). They implemented a simulation of dBM reduction for those groups, assuming access to *a perfect* CDH *oracle* that solves the CDH problem with probability 1. In such a setting, they managed to break the DL of many groups in around 30 seconds on modest hardware while requiring a few terabytes of memory.

This is a good indication that DL and CDH might be computationally equivalent assumptions in many groups. However, there is also a significant caveat regarding non-idealized reductions. In a real reduction, we assume access to an imperfect ε-CDH oracle that runs at most in time t and breaks the CDH with probability ε. In a non-idealized case, the reduction would have to amplify the ε-CDH oracle to a near-perfect CDH oracle. Shoup [Sho97] showed that one can simulate $(1-\beta)$-CDH oracle for an arbitrarily small $\beta \in (0,1)$ by making $\mathcal{O}(\log(1/\beta)/\varepsilon)$ calls to a ε-CDH oracle. Therefore, if the idealized reduction requires c perfect oracle calls, the non-idealized one will have an additive overhead of at least $\mathcal{O}(t \cdot c \cdot \log(1/\beta)/\varepsilon)$ in the running time compared to the idealized reduction. Thus, it is essential to minimize c. The reductions in [MS23] have $c > 10,000$.

Furthermore, many modern cryptographic protocols do not rely on just the DL or CDH assumption but on some generalization of those. In the literature, one can find hundreds of variations and extensions of the DL and CDH assumptions. See [Boy08, MRV16, GG17] for some well-known families of such assumptions. Typically, little is known beyond the fact that they are hard to break in the idealized group models. Currently, the most popular of these idealized group models are the generic group model (GGM) [Sho97, Mau05] and the algebraic group model (AGM) [FKL18, LPS23, JM24]. The generic group model hides the underlying structure of the group by either representing group elements with a random injective encoding function [Sho97] or by abstract memory handles [Mau05]. The adversary can operate with the group elements through oracles, which perform the group operation, exponentiation, and equality test. This guarantees that the adversary's algorithm could be run on any abstract group.

On the other hand, the algebraic group model reveals the group structure to the adversary but still requires that the adversary can produce new group elements only through generic group operations. A significant consequence of this is that the security proofs become security reductions. In the generic group model, the security proofs are typically unconditional. They show that a generic adversary must perform a superpolynomial number of operations to break the assumption. Many security proofs [FKL18, BFL20, Rot22, TZ23] in the algebraic group model show that the hardness of the n-Power Discrete Logarithm (n-PDL) assumption implies the hardness of the relevant assumption or cryptographic protocol. Recall, n-PDL is a generalization of the DL assumption where the adversary gets as an input $g, g^x, g^{x^2}, \ldots, g^{x^n}$ for a random exponent x and has to output x. Bauer, Fuchsbauer, and Loss [BFL20] show that even the Uber assumption [Boy08], a family for a wide variety of DL-based assumptions, is hard under the n-PDL assumption. As a relevant example, we consider in this work the n-Diffie-Hellman Exponent (n-DHE) assumption stating that it is hard to compute $g^{x^{n+1}}$ on input $g, g^x, g^{x^2}, \ldots, g^{x^n}$. It is equivalent to the Diffie-Hellman Inversion assumption [BBG05], where the adversary gets the same input but has to output $g^{1/x}$, which has been used in various works [ZSS04, BB04a, BBG05, Che06, Cam13, CF13, DGNW20]. The inversion problem is also a special case of the n-Strong Diffie-Hellman (SDH) assumption, which has had notable impact on pairing-based cryptography [BB04b, BB04a, Gen04, KZG10, PHGR13, LPS24].[3]

However, the analysis in the idealized group models should only be considered a security heuristic. Cryptanalysis algorithms, such as index calculus, are non-generic algorithms.

[3] In n-SDH assumption, the adversary obtains $g, g^x, g^{x^2}, \ldots, g^{x^n}$ and has to output $g^{1/(x+c)}$ and a constant c. The case $c = 0$, is equivalent to n-DHE.

Even the additive group $\mathbb{Z}_p$ does not have an efficient generic discrete logarithm, although computing discrete logarithms is trivial in that group. Although the same arguments do not directly apply to the algebraic group model, we also know of primitives that are secure in the AGM but insecure with any standard model instantiation of a group [Zha22]. Similar counterexamples are also known for both Shoup's [Den02] and [Zha22] formalizations of the generic group model.

Clarifying the hardness of DL-based assumptions in the standard model remains an open problem in cryptography. Moreover, the reduction technique proposed by den Boer and Maurer is essentially the only (non-trivial) available tool. This motivates us to ask:

> *Can we apply* dBM *reduction technique to analyze other assumptions in the discrete logarithm setting?*

Our Contribution.

To answer this question, we revisit the dBM reduction, focusing on groups of prime order p, where $p-1$ is somewhat smooth, e.g., every prime q that divides $p-1$ is less than 2^{100}. As a result, we present the following contributions:

1. We provide a generalized framework for the dBM reduction that solves n-PDL, instead of DL, when given access to a challenge exponentiation oracle Exp. If one shows that an oracle for solving X is sufficient to build Exp, the n-PDL assumption implies the assumption X. This realization allows the dBM technique to be applied to other assumptions. Importantly, we show that an oracle for n-DHE can implement such exponentiation oracle efficiently when $\mathsf{n} = \mathsf{poly}(\lambda)$, where λ is the security parameter. Thus, n-PDL assumption implies the n-DHE in this case.
2. On the impossibility front, we show that the current framework (or any other generic reduction) cannot reduce a family of Generalized DHE (GDHE) [GG17] assumptions to any of the PDL assumptions. This contrasts with the algebraic group model, where such reductions are possible, and highlights the limitations of the dBM technique.
3. Additionally, we show that our approach is relevant for important classes of elliptic curves, for instance, BLS12-381. We see that DL and CDH are essentially equally hard to break in BLS12-381. The running time of our reduction is significantly faster than any prior work.
4. Lastly, for the sake of simplicity, reductions in this framework assume perfect oracles for the hard tasks (i.e., CDH or DHE). We also show a practical and efficient way to implement these oracles, given adversaries that succeed with probability less than 1.

1.1 Technical Overview

We recall the main idea behind the dBM reduction. Let $\mathbb{G}$ be an additive cyclic group of prime order p. We denote the generator of $\mathbb{G}$ by $[\![1]\!]$ and $[\![x]\!] := x \cdot [\![1]\!]$ for any $x \in \mathbb{F}_p$.

Den Boer, Maurer, and Wolf [den90, Mau94, MW96, MW99] showed how to compute discrete logarithms in $\mathbb{G}$, when given access to a CDH oracle and a suitable auxiliary group with a smooth order. Recall that a number is called α-smooth if each of its prime divisors p_i is bounded by 2^α and it is called α-powersmooth if each of its prime power divisor $p_i^{e_i}$ is bounded by 2^α. The most common choice for an auxiliary group is an elliptic curve $E(\mathbb{F}_p)$ with either a smooth or a powersmooth order, depending on the exact algorithm.

Suppose $\mathcal{P}$ denotes a generator $E(\mathbb{F}_p)$. Since the reduction can perform both additions and multiplications (the latter with the CDH oracle) inside the brackets, it can, in fact, perform all generic $\mathbb{F}_p$ operations on x. The reduction maps the DL challenge element $[\![x]\!]$ to $[\![x, y]\!]$ such that (x, y) is an element of the elliptic curve $E(\mathbb{F}_p)$. We call $[\![x, y]\!]$ an implicit representation of $(x, y) \in E(\mathbb{F}_p)$. Using the CDH oracle, the reduction can perform elliptic curve additions and scalar multiplications on the implicit representation. It can

also perform equality tests on the implicit representations to test whether two elliptic curve points are the same. This set of operations is sufficient to run some generic discrete logarithm algorithm, such as the Silver-Pohlig-Hellman algorithm [PH78], on the implicit representation $[\![x, y]\!]$. These algorithms are efficient when the order of $E(\mathbb{F}_p)$ is smooth. Namely, they run in time $\mathcal{O}(2^\alpha)$ when $E(\mathbb{F}_p)$ has an α-smooth order.

Hence, the reduction can compute u such that $u \cdot [\![\mathcal{P}]\!] = [\![x, y]\!]$ for some known elliptic curve generator point $\mathcal{P}$. Once we know u, one can compute $u \cdot \mathcal{P} = (x, y)$ and recover x.

Generalization for Smooth $p-1$. Subsequent works have focused on instantiating the dBM reduction with a smooth elliptic curve [MSV04, Ben05, MS23]. However, den Boer [den90] observed that one can use the finite field's multiplicative group $\mathbb{F}_p^*$ as an auxiliary group. The main challenge with this approach is that $|\mathbb{F}_p^*| = p - 1$ is not always going to be sufficiently smooth.

We start by observing that several highly influential elliptic-curve groups have a $p-1$ which is α-smooth. The values of α are represented in Table 1. In particular, many SNARK-friendly pairing groups intentionally choose $p-1$ such that it is divisible by a large power of 2 due to efficiency reasons. This results in many groups from the BLS family [BLS03] and BN family [BN06, PSNB11] having a smooth $p-1$. The most interesting of those pairing groups is BLS12-381 [BGM17], which has an exceptionally smooth $p-1$ (it is 28-smooth). BLS12-381 has become the default pairing choice in recent years. Various blockchain protocols have adopted BLS12-381 for their BLS signature scheme [Net24, Eth24] and ZK-SNARKs [Con24, Sui24]. BLS12-381 will also be part of Ethereum's precompiled contracts (native operations) in an upcoming update[4]. However, non-pairing groups NIST P-256 and secp256k1 also have a relatively smooth $p-1$. The latter two are some of the most widely used elliptic curves on the internet due to OpenSSL and Bitcoin.

Table 1: The α-smoothness of $p - 1 = |\mathbb{F}_p^*|$ for several elliptic curve groups of order p

	Curve											
	BLS12-							BN-			NIST P-	secp256k1
	381	455	377	466	638	317	477	158	190	254	256	
$\boldsymbol{\alpha}$	28	48	64	73	97	40	42	68	46	100	92	109

We present a generalization of den Boer's version of the dBM reduction, where $\mathbb{F}_p^*$ is the auxiliary group. The reduction takes as an input $[\![1, x, \ldots, x^{\mathsf{n}}]\!]$ and it is equipped with an exponentiation oracle Exp that given any integer m returns $[\![x^m]\!]$. This is a generalization in two different ways:

1. The reduction solves n-PDL assumption instead of the usual discrete logarithm assumption. The discrete logarithm assumption is obtained for $\mathsf{n} = 1$.
2. The oracle performs exponentiations, which we identify as crucial for the dBM approach to work. One can trivially realize Exp oracle with a CDH oracle. However, later, we also show that another established assumption (n-DHE) is sufficient to realize Exp oracle. Conceptually, our framework hides the complexity of the dBM reduction and highlights that if an oracle for an assumption X can implement Exp, then n-PDL implies the assumption X. This allows us to apply the underlying technique of the dBM reduction to other assumptions.

Our use of $\mathbb{F}_p^*$ as the auxiliary group significantly reduces the number of Exp queries needed since we avoid expensive elliptic curve scalar multiplications.

[4] See https://eips.ethereum.org/EIPS/eip-2537.

Suppose $p-1=\prod_{i=1}^{d} p_i^{e_i}$ is the prime factorization. We show that it is only necessary to use the oracle to compute $[\![x^{(p-1)/p_i^k}]\!]$ for all $i=1,\dots,d$ and $k=1,\dots,e_i$. This requires only $\sum_{i=1}^{d} e_i$ queries to the Exp oracle, where $e_1,\dots,e_d$ and d are typically some small constants. The memory requirement is $\mathcal{O}\big(\sqrt{p_i}\log p\big)$ for the largest prime p_i and the number of group/field operations is $\mathcal{O}\Big(\sum_{i=1}^{d}\sqrt{p_i}\Big)$.

As a concrete example, let us consider BLS12-381, which has a particularly low 28bit smoothness. We show that our reduction requires 1761 CDH queries, around 6MB of memory, and around 60,000 finite field multiplications and scalar multiplication in Section 3.2. The same group was also considered in recent work by May and Schneider [MS23], where they found an auxiliary elliptic curve with 36bit smoothness. May and Schneider report 20,397 CDH queries, 1.9TB of memory, and a running time of 28s. Thus, we have shown that $\mathsf{DL}=\mathsf{CDH}$ in BLS12-381 with a relatively good reduction tightness. In fact, our reduction for BLS12-381 seems to be by far the most efficient instantiation of the dBM reduction for any practically relevant group.

n-PDL $\Rightarrow$ n-DHE. To establish the notation, we write $X \Rightarrow Y$, when the hardness of solving the assumption X implies the hardness of solving the assumption Y. See Section 2.1.

The main goal of our framework, however, is not to outperform the recent work of [MS23] (for many relevant groups, they are likely more efficient) but to apply the dBM reduction to new assumptions. This is significantly easier when the auxiliary group is $\mathbb{F}_p^*$ since the oracle only needs to compute $[\![x^m]\!]$ for a small number of different values of m as we explained above. However, m may be of exponential size in the security parameter.

We show in Section 3.3 that the n-Power Discrete Logarithm (n-PDL) assumption implies n-Diffie-Hellman Exponent (n-DHE) assumption in groups with (somewhat) smooth $p-1$ such as BLS12-381. Recall, in the n-PDL assumption, the adversary gets $[\![1,x,x^2,\dots,x^{\mathsf{n}}]\!]$ as an input and has to compute x, and in the n-DHE assumption, the adversary gets the same input, but has to compute $[\![x^{\mathsf{n}+1}]\!]$. n-PDL and n-DHE can be seen as generalizations of DL and CDH assumptions ($\mathsf{DL}=1$-PDL and $\mathsf{CDH}=1$-DHE).

To achieve the reduction, we need to realize our framework's Exp oracle using a polynomial number of n-DHE oracle queries. However, this is not straightforward since the n-DHE oracle requires an input of the form $([\![\mathsf{g}]\!],[\![y\mathsf{g}]\!],\dots,[\![y^{\mathsf{n}}\mathsf{g}]\!])$ for some generator $\mathsf{g}\in(\mathbb{F}_p,+)$ and $y\in\mathbb{F}_p$. Hence, we present an iterative algorithm to efficiently compute $[\![x^m]\!]$ based on modular arithmetic. The idea is to construct finite lists with the initial terms of arithmetic progressions that contain the exponent m and expand these lists on each iteration until we recover m. More specifically, we start with a list $L=[\![1,x,x^2,\dots,x^{\mathsf{n}}]\!]$ and calculate the $\mathsf{n}+1$ consecutive terms of the list by $\mathsf{n}+1$ calls to $\mathsf{O}_{\mathrm{dhe}}(L)$.

Then, in the first iteration, we calculate $a\equiv m \bmod 2$ and set $d=2$ in an arithmetic progression, to recover the $\mathsf{n}+1$ terms $[\![x^a,x^{a+2},\dots,x^{a+2\mathsf{n}}]\!]$ from L. Afterwards, with $\mathsf{n}+1$ calls to $\mathsf{O}_{\mathrm{dhe}}(\cdot)$, we obtain $L_1=[\![x^a,x^{a+2},\dots,x^{a+2\mathsf{n}},x^{a+(\mathsf{n}+1)2},\dots,x^{a+(2\mathsf{n}+1)2}]\!]$. Proceeding iteratively, with $a=m \bmod 2^i$ and $d=2^i$, and updating the list at each iteration i, we will recover $[\![x^m]\!]$ after $\mathcal{O}((\mathsf{n}+1)\log(m/\mathsf{n})$ calls to the n-DHE oracle. More details can be found in Section 3.3.

Impossibility Results. We observe that the dBM reduction is a generic reduction. It performs only generic operations with the group elements. We prove two impossibility results for generic reduction algorithms, which show that there are significant limitations to the dBM technique in Section 4.

1. In the n-$\mathsf{PDL}\Rightarrow\mathsf{n}$-$\mathsf{DHE}$, we assume that the reduction can call the n-DHE oracle on an arbitrary input $([\![\mathsf{g}]\!],[\![y\mathsf{g}]\!],\dots,[\![y^{\mathsf{n}}\mathsf{g}]\!])$. In particular, the reduction could pick a new group generator $[\![g]\!]$ on every call. This is not perfectly aligned with practice, where the cryptographic library often fixes the group generator once and for all. It

raises the question if changing the generator could be avoided in the reduction. We show that there is no generic reduction algorithm given oracle access to the m-DHE solver with a fixed-base generator, which breaks the n-PDL assumption for $\mathsf{n} < 2m$. We enforce fixed-base by having the DHE oracle verify that the generator $[\![g]\!]$ is equal to a global generator $[\![1]\!]$ and that the input is well-formed. That is, the input must have the structure $([\![1]\!], [\![y]\!], \ldots, [\![y^m]\!])$ for some integer y.

2. Our success with showing n-PDL $\Rightarrow$ n-DHE might cause optimism that some form of Uber assumption [Boy08, GG17] is implied by n-PDL. This is known to be true in the algebraic group model [BFL20]. Furthermore, Ghadafi and Groth [GG17] showed that a certain form of target Uber assumption is implied by n-GDHE (Generalized Diffie-Hellman) and n-SFrac (Simple Fractional) assumptions [5]. The first of these is a generalization of n-DHE and states that given $[\![1, x, \ldots, x^{\mathsf{n}-1}, x^{\mathsf{n}+1}, \ldots, x^{2\mathsf{n}}]\!]$ it is hard to compute $[\![x^{\mathsf{n}}]\!]$. In particular, n-GDHE $\Rightarrow$ $(\mathsf{n}-1)$-DHE. We show that for $m \geq 2, \mathsf{n} \geq 1$, there is no efficient generic reduction, which would allow us to conclude n-PDL $\Rightarrow$ m-GDHE. That is n-PDL $\overset{\text{gen}}{\not\Rightarrow}$ m-GDHE. This provides an interesting contrast with the algebraic group model, where 2n-PDL $\overset{\text{alg}}{\Longrightarrow}$ n-GDHE.

The proofs of both results rely on the fact that given the GDHE oracle with a variable base generator or DHE oracle with a fixed base generator, it is impossible to compute $[\![f(x)]\!]$ for a polynomial f of superpolynomial degree in the security parameter. Moreover, we use the standard generic group model proof strategy [Sho97], which boils down to showing that for any two distinct polynomials $f_i(X)$ and $f_j(X)$, $f_i(x) \neq f_j(x)$ with an overwhelming probability, for the challenge exponent x. When the degree of the polynomials has a polynomial size in the security parameter, it clearly holds since x is sampled from a superpolynomial size set. These proofs provide an interesting insight into the dBM reduction or any other similar generic reduction. The ability to compute high-degree polynomials in the challenge value x is essential for the generic model reduction to succeed.

The Use of Perfect Oracles. So far, we considered perfect oracles that solve computational tasks with probability 1, similarly to what is done in [MW96, MS23]. For instance, in Section 3.3 we consider an oracle for DHE that always returns the correct answer. However, in principle, this is not enough to claim that PDL $\Rightarrow$ DHE. In fact, if the DHE assumption does not hold, then there exists an efficient adversary $\mathcal{A}$ that solves DHE with a non-negligible yet possibly relatively small probability. Furthermore, the computational tasks we consider are not publicly verifiable. Consider the CDH assumption for simplicity. Only a challenger who knows the secrets x, y used to generate the input $[\![x, y]\!]$ can check if the adversary returned the correct answer. Since the reduction does not have a secret, it cannot directly check whether the result from the oracle is correct. A similar issue occurs for the DHE oracle. Thus, one cannot rely on the naive strategy of defining the perfect oracle by calling the adversary $\mathcal{A}$ until it gets the correct answer.

To circumvent this issue, we resort to a more sophisticated method of amplifying the success probability of an adversary that solves a privately verifiable, computational task. In Section 5.1, we show how to define an efficient adversary that solves CDH with probability arbitrarily close to 1, given oracle access to an efficient adversary $\mathcal{A}$ that solves CDH with non-negligible probability. The construction is inspired by similar results in literature [Sho97, MW96]. Maurer [MW96] proposes an amplification technique, but it is not asymptotically optimal. Shoup's [Sho97] technique is asymptotically tight but assumes prior knowledge of the success probability ε of the imperfect oracle, and its efficiency is analyzed only asymptotically. Our solution is asymptotically as efficient as [Sho97] and does not require knowledge of ε. See Section 5 for a detailed comparison between our

[5] n-SFrac states that given $[\![1, x, \ldots, x^{\mathsf{n}}]\!]$ it is hard to compute $[\![r(x)/s(x)]\!]$ and polynomials $r(X)$, $s(X)$ such that $0 \leq \deg(r) < \deg(s) \leq \mathsf{n}$. We leave the analysis of SFrac as an open problem.

amplification technique and the previous ones. Finally, in Section 5.2, we show how to adapt the result for the case of DHE. We define an efficient adversary that solves DHE with probability arbitrarily close to 1, given oracle access to an efficient adversary $\mathcal{A}$ that solves DHE with non-negligible probability.

We present a more detailed related work in Appendix A.

2 Preliminaries

Let λ denote the security parameter, and PPT stands for probabilistic polynomial time Turing machine. Sampling a uniformly from a set A is denoted by $a \leftarrow\!\!\$\ A$. Let $\mathbb{F}_p$ denote a finite field of prime order p and $\mathbb{F}_p^* := \mathbb{F}_p \setminus \{0\}$ its multiplicative group. For $a, b \in \mathbb{Z}$, $a \leq b$, we write $[a : b] := \{a, a+1, \ldots, b\}$. We denote the natural logarithm of x by $\ln(x)$ or $\ln x$, and the logarithm of x to the base 2 by $\log(x)$ or simply $\log x$.

Let Ggen be a group generator that takes as an input 1^λ and outputs $(\mathbb{G}, \mathcal{P}, p)$, where $\mathbb{G}$ is an additive cyclic group of prime order p and $\mathcal{P}$ is its generator. In the following we will denote $[\![1]\!] := \mathcal{P}$ and $[\![x]\!] := x[\![1]\!]$. This notation also extends to vectors $[\![x_1, \ldots, x_n]\!] := ([\![x_1]\!], \ldots, [\![x_n]\!])$.

2.1 Reduction Notation

If there exists a PPT standard model reduction, which shows that the hardness of an assumption X implies the hardness of an assumption Y, we write $X \Rightarrow Y$. For example, $\mathsf{CDH} \Rightarrow \mathsf{DL}$. If there exists no PPT reduction algorithm that would show $X \Rightarrow Y$, then we write $X \not\Rightarrow Y$. If the reduction is a generic (respectively algebraic) algorithm, we write $X \overset{\text{gen}}{\Longrightarrow} Y$ (respectively $X \overset{\text{alg}}{\Longrightarrow} Y$).

2.2 Assumptions

The following assumption is often used jointly with the algebraic group model.

Definition 1 (n-PDL)**.** We say that the n-Power Discrete Logarithm assumption is secure respect to Ggen if for any PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathrm{pdl}}_{\mathsf{Ggen},\mathcal{A},\mathsf{n}}(\lambda) := \Pr\left[x = \mathcal{A}(\mathsf{p}, [\![1, x, x^2, \ldots, x^{\mathsf{n}}]\!]) \mid \mathsf{p} \leftarrow \mathsf{Ggen}(1^\lambda), x \leftarrow\!\!\$\ \mathbb{F}_p\right] \approx_\lambda 0 \ .$$

When taking $\mathsf{n} = 1$, we obtain the standard DL assumption. Next, we recall the Computational Diffie-Hellman assumption.

Definition 2 (CDH)**.** We say that the Computational Diffie-Hellman (CDH) assumption is secure respect to Ggen if for any PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathrm{cdh}}_{\mathsf{Ggen},\mathcal{A}}(\lambda) := \Pr\left[[\![xy]\!] = \mathcal{A}(\mathsf{p}, [\![1, x, y]\!]) \mid \mathsf{p} \leftarrow \mathsf{Ggen}(1^\lambda), x, y \leftarrow\!\!\$\ \mathbb{F}_p\right] \approx_\lambda 0 \ .$$

One possible generalization of this assumption is the n-Diffie-Hellman Exponent (n-DHE) assumption [ZSS04].

Definition 3 (n-DHE)**.** We say that the (variable base) n-Diffie-Hellman Exponent assumption is secure respect to Ggen if for any PPT adversary $\mathcal{A}$, and any generator $\mathsf{g} \in (\mathbb{F}_p, +)$,

$$\mathsf{Adv}^{\mathrm{dhe}}_{\mathsf{Ggen},\mathcal{A},\mathsf{n}}(\lambda) := \Pr\left[\ [\![x^{\mathsf{n}+1}\mathsf{g}]\!] = \mathcal{A}(\mathsf{p}, [\![\mathsf{g}, x\mathsf{g}, x^2\mathsf{g}, \ldots, x^{\mathsf{n}}\mathsf{g}]\!]) \ \middle|\ \begin{array}{c} \mathsf{p} \leftarrow \mathsf{Ggen}(1^\lambda), \\ x \leftarrow\!\!\$\ \mathbb{F}_p \end{array} \ \right] \approx_\lambda 0 \ .$$

We say that the above assumption has a fixed base if $\mathsf{g} = 1$ and thus one uses the generator $[\![1]\!]$ determined by $\mathsf{p} \leftarrow \mathsf{Ggen}(1^\lambda)$. The 1-DHE assumption is often called the Square Computational Diffie-Hellman (SCDH) assumption, and it is known to be equivalent to the CDH assumption [MW96]. We sketch a proof in Appendix B.1.

$\mathsf{Enc}(y)$	$\mathsf{Add}(\xi, \xi', b)$	$\mathcal{I}(\xi)$
$t := \lvert\vec{\sigma}\rvert$;	**if** $\xi \notin \vec{\sigma} \vee \xi' \notin \vec{\sigma}$ **then**	$i := \min\{k \in [1 : \lvert\vec{\sigma}\rvert] : \xi = \sigma_k\}$;
if $\exists i \in [1 : t] : y = x_i$ **then**	**return** $\perp$; **endif**	**return** i;
$\sigma_{t+1} := \sigma_i$;	$i := \mathcal{I}(\xi); j := \mathcal{I}(\xi)$;	
else	$y := x_i + (-1)^b x_j$;	
$\sigma_{t+1} \leftarrow\!\!\$\ \{0,1\}^{\log p} \setminus \{\sigma_j\}_{j=1}^{t}$;	**return** $\mathsf{Enc}(y)$;	
$\vec{x} := \vec{x} \Vert y; \vec{\sigma} := \vec{\sigma} \Vert \sigma_{t+1}$;		
return σ_t;		

Figure 1: The GGM functions Enc, Add, and $\mathcal{I}$. Initially $\vec{x} = ()$ and $\vec{\sigma} = ()$.

2.3 Generic Group Model

The generic group model (GGM) [Sho97] is an idealized model for analyzing assumptions and protocols in the DL-related setting. Here, we use Shoup's version of the GGM [Sho97]. Maurer proposed an alternative formulation of the GGM [Mau05], and also, the algebraic group model [FKL18] is often used similarly. See [Zha22] for their comparison.

In Shoup's GGM, group elements of a prime order p group are modeled by a random injective function $\mathsf{Enc} : \mathbb{F}_p \to \{0,1\}^{\lceil \log p \rceil}$. We implement it with lazy sampling. The security game initializes the input vector $\vec{x}$ and the encoding vector $\vec{\sigma}$. When some element $y \in \mathbb{F}_p$ needs a group element encoding, the algorithm $\mathsf{Enc}(y)$ checks if $y = x_i \in \vec{x}$ and if so, it returns the i-th encoding σ_i from $\vec{\sigma}$. If y does not have an encoding yet, Enc samples $\sigma_{|\vec{\sigma}|+1} \leftarrow\!\!\$\ \{0,1\}^{\lceil \log p \rceil} \setminus \{\sigma_i\}_{i=1}^{|\vec{\sigma}|}$. Whether a new encoding had to be created or not, Enc appends y to the vector $\vec{x}$ and its encoding to the vector $\vec{\sigma}$. See $\mathsf{Enc}(\cdot)$ algorithm in Fig. 1.

A generic algorithm typically gets encodings of some group elements $\mathsf{Enc}(1)$, $\mathsf{Enc}(a_1)$, ..., $\mathsf{Enc}(a_\mathsf{n})$ as an input. The only way it can operate with the group elements is through an addition oracle Add. The Add oracle takes two encodings ξ and ξ' as an input. It then looks up x_i and x_j with the smallest index i and j (we model it with the function $\mathcal{I}$ in Fig. 1) from the vector $\vec{x}$ corresponding to ξ and ξ' and returns $\mathsf{Enc}(x_i + x_j)$. If either of the encodings ξ or ξ' is not in $\vec{\sigma}$, then Add will not perform the operation and returns $\perp$. See the second algorithm in Fig. 1. Occasionally a generic algorithm will have access to more oracles. We discuss those cases later in the paper.

The algebraic group model (AGM) [FKL18] is a different ideal model for cryptographic groups. See Appendix B.2 for a description of AGM.

3 Generalizing dBM Reduction

Let $\mathbb{G}$ be an additive group of prime order p and $[\![1]\!]$ its generator. We aim to generalize dBM reduction and show that n-PDL $\Rightarrow$ n-DHE. As mentioned in the introduction, we assume for simplicity that the order of $\mathbb{F}_p^*$ is (somewhat) smooth as was considered in the original work by den Boer [den90]. This is, for example, satisfied by the popular pairing group BLS12-381, which has $p - 1$ with exceptionally low 28-bit smoothness (see Table 1).

In this section, we describe a reduction $\mathcal{A}_{\mathsf{PDL}}$ for breaking the n-PDL assumption when equipped with an exponentiation oracle Exp. The algorithm $\mathcal{A}_{\mathsf{PDL}}$ gets as an input a PDL challenge $[\![1, x, \ldots, x^n]\!]$ and it has to output x. To achieve this, $\mathcal{A}_{\mathsf{PDL}}$ has access to an oracle Exp that takes as an input any integer m and outputs $[\![x^m]\!]$. Our $\mathcal{A}_{\mathsf{PDL}}$ generalizes dBM reduction, which is originally equipped with a multiplication oracle (given any $[\![a]\!]$ and $[\![b]\!]$ the oracle outputs $[\![ab]\!]$). We see that the more restrictive exponentiation oracle, which can only exponentiate the challenge element x, is sufficient. In Section 3.3, we show how to realize the Exp oracle by having access to a n-DHE oracle.

3.1 Description

We proceed by explaining our algorithm $\mathcal{A}_{\mathsf{PDL}}^{\mathsf{Exp}(\cdot)}$, described in Fig. 2.

Finding a Generator of $\mathbb{F}_p^*$. Suppose that the order $p-1$ of the multiplicative group $\mathbb{F}_p^*$ factorizes as $p-1=\prod_{i=1}^{d} p_i^{e_i}$, where p_i are distinct primes. It is well-known that $\mathbb{F}_p^*$ is a cyclic group. Also, every subgroup of a cyclic group is cyclic, and for every $s \mid (p-1)$, there is precisely one subgroup of $\mathbb{F}_p^*$ of order s. We denote this subgroup by $\mathbb{H}_s$.

As a first step, $\mathcal{A}_{\mathsf{PDL}}$ finds a generator of $\mathbb{F}_p^*$. This is done by picking a random $g \leftarrow\!\!\$\ \mathbb{F}_p^*$ and testing if $g^{(p-1)/p_i} \not\equiv 1 \pmod p$ for $i \in [1:d]$. If yes, then g is a generator of $\mathbb{F}_p^*$; otherwise, we resample g and repeat the procedure until the condition holds. For relevant sizes of p, the next lemma shows that it takes only a small number of iterations. We give the proof in Appendix C.

Lemma 1. *Let X be the random variable indicating the number of iterations until a randomly sampled $g \leftarrow\!\!\$\ \mathbb{F}_p^*$ is a generator of $\mathbb{F}_p^*$. For any prime $p \geq 2^{43}$, expected value $\mathbb{E}[X] < 2\ln\ln(p-1)$.*

Later on we will use the fact that $h_i := g^{(p-1)/p_i^{e_i}}$ is a generator of $\mathbb{H}_{p_i^{e_i}}$.

Solving PDL in the Subgroup. Suppose $[\![1, x, \ldots, x^{\mathsf{n}}]\!]$ is the n-PDL challenge and $p-1=\prod_{i=1}^{d} p_i^{e_i}$ as defined earlier. Let $y \in \mathbb{F}_p^*$ be such that $[\![x]\!] = [\![g^y]\!]$ (assuming $x \neq 0$). Observe that if we find y, we have also recovered $g^y \bmod p = x$.

We fix a $q = p_i$ and $e = e_i$ for some $i \in [1:d]$. Recall, $h := g^{(p-1)/q^e}$ is the generator of the subgroup $\mathbb{H}_{q^e}$ of $\mathbb{F}_p^*$. The algorithm SubPDL in Fig. 2 follows the well-known strategy of Silver-Pohlig-Hellman algorithm to recover $y' \in [0 : q^e - 1]$ that satisfies $y \equiv y' \pmod{q^e}$. We explain it below.

Let us express y' in the q-adic form as

$$y' = y'_0 + y'_1 q + \cdots + y'_{e-1} q^{e-1} \ , \tag{1}$$

where each $y'_i \in [0 : q-1]$ and $0 \leq i \leq e-1$. It is possible to project x to the subgroup $\mathbb{H}_{q^e}$ in the following sense,

$$x' := x^{(p-1)/q^e} \equiv (g^y)^{(p-1)/q^e} \equiv h^y \overset{(a)}{\equiv} h^{y'} \equiv \prod_{i=0}^{e-1} h^{y'_i q^i} \pmod p \ , \tag{2}$$

and (a) holds from the equivalence $y \equiv y' \pmod{q^e}$.

First, SubPDL computes $\gamma := h^{q^{e-1}} \pmod p$. Since $\gamma^q \equiv h^{q^e} \equiv 1 \pmod p$, γ belongs to the subgroup $\mathbb{H}_q$. Moreover, since $\gamma \neq 1$ (otherwise h would not be a generator of $\mathbb{H}_{q^e}$) and the order of γ has to divide the prime q, γ itself must have the order q. That makes γ a generator $\mathbb{H}_q$. Then,

$$x^{(p-1)/q} \equiv (x^{(p-1)/q^e})^{q^{e-1}} \equiv (x')^{q^{e-1}} \overset{(b)}{\equiv} \gamma^{y'_0} \cdot \prod_{i=1}^{e-1} h^{y'_i q^{e-1+i}} \overset{(c)}{\equiv} \gamma^{y'_0} \pmod p \ ,$$

where (b) follows from Eq. (2) and (c) from $h^{q^e} \equiv 1 \pmod p$.

SubPDL calls the Exp oracle to compute $\mathsf{Exp}((p-1)/q) = [\![\gamma^{y'_0}]\!]$, and then recovers y'_0 by using Baby-Step Giant-Step (BSGS) algorithm to compute DL in $\mathbb{H}_q$. Next, observe that $x' h^{-y'_0} \equiv h^{y'_1 q + \cdots + y'_{e-1} q^{e-1}} \pmod p$ and therefore

$$t := (x' h^{-y'_0})^{q^{e-2}} \equiv \gamma^{y'_1} \cdot h^{y'_2 q^e} \cdot \ldots \cdot h^{y'_{e-2} q^{2e-3}} \equiv \gamma^{y'_1} \pmod p \ .$$

$\mathcal{A}_{\mathsf{PDL}}^{\mathsf{Exp}(\cdot)}([\![1, x, \ldots, x^{\mathsf{n}}]\!])$

if $[\![x]\!] = [\![0]\!]$ **do**
  return 0; **fi**
$g \leftarrow\!\$\ \mathbb{F}_p^*$; // Find a generator of $\mathbb{F}_p^*$
while $\exists i \in [1:d] : g^{(p-1)/p_i} \bmod p = 1$
  $g \leftarrow\!\$\ \mathbb{F}_p^*$;
endwhile
for $i \in [1:d]$ **do**
  $y_i := \mathsf{SubPDL}^{\mathsf{Exp}(\cdot)}([\![1, x]\!], g, p, p_i, e_i)$;
endfor
Solve the following system for $y \in \mathbb{F}_p$:

$$\begin{cases} y \equiv y_1 \pmod{p_1^{e_1}} \\ \vdots \\ y \equiv y_d \pmod{p_d^{e_d}} \end{cases}$$

return $g^y \bmod p$;

$\mathsf{SubPDL}^{\mathsf{Exp}(\cdot)}([\![1, x]\!], g, p, q, e)$

Initialize a hash table H;
$h := g^{(p-1)/q^e}$; $\gamma := h^{q^{e-1}}$; $z := 1$;
for $v \in [0 : \lceil\sqrt{q}\rceil - 1]$ **do**
  Compute $[\![\gamma^v]\!] = z \cdot [\![1]\!]$;
  Set $H([\![\gamma^v]\!]) = v$;
  Compute $z := \gamma \cdot z \bmod p$; // $z = \gamma^{v+1} \bmod p$
endfor
$y := 0$;
for $k \in [0 : e-1]$ **do**
  $[\![t']\!] := \mathsf{Exp}((p-1)/q^{k+1})$;
  Compute $[\![t]\!] := h^{-yq^{e-1-k}} \cdot [\![t']\!]$;
  for $u \in [0 : \lceil\sqrt{q}\rceil - 1]$ **do**
    Find $v := H(\gamma^{-u\lceil\sqrt{q}\rceil} \cdot [\![t]\!])$;
    if $v \neq \perp$ **then break**;
  endfor
  Set $y' = u\lceil\sqrt{q}\rceil + v$;
  Set $y := y + q^k \cdot y'$;
endfor
return y;

Figure 2: Algorithm $\mathcal{A}_{\mathsf{PDL}}$ for solving the PDL problem and its subalgorithm SubPDL. Algorithms are equipped with an oracle Exp that outputs $[\![x^m]\!]$ on an input m. Here the group has an order p and $p - 1 = \prod_{i=1}^{d} p_i^{e_i}$ is smooth prime factorization.

Given that $(x')^{q^{e-2}} \equiv x^{(p-1)/q^2} \pmod p$, SubPDL computes $\mathsf{Exp}(^{(p-1)}/_{q^2}) = [\![x^{(p-1)/q^2}]\!]$ and then $[\![t]\!] = (g^{-y_0' q^{e-2}}) \cdot [\![x^{(p-1)/q^2}]\!] = [\![\gamma^{y_1'}]\!]$. Again, the algorithm can use BSGS to compute y_1 in $\mathbb{H}_q$. Next it computes $[\![t]\!] := [\![(x' \cdot h^{-(y_0' + y_1' q)})^{q^{e-3}}]\!] = [\![\gamma^{y_2'}]\!]$ and continues analogously until $y_0', \ldots, y_{e-1}'$ are recovered. Finally, SubPDL uses Eq. (1) to compute y'.

As mentioned above, we use BSGS algorithm to compute DL in $\mathbb{H}_q$. In SubPDL, we precompute $[\![\gamma^u]\!]$ for all $u \in [0 : \lceil\sqrt{q}\rceil - 1]$ and construct a hash map H which maps $[\![\gamma^u]\!]$ to u. Recall that there are hash maps with insertion and look-up of $\mathcal{O}(1)$ complexity. Storing the hash table requires $\mathcal{O}\big(\sqrt{q} \cdot \log p\big)$ memory, and constructing the table takes $\mathcal{O}\big(\sqrt{q}\big)$ scalar multiplications in $\mathbb{G}$. BSGS is based on the idea that y_i' can be uniquely expressed as $y_i' = u\lceil\sqrt{q}\rceil + v$, where $0 \leq u, v \leq \lceil\sqrt{q}\rceil - 1$. SubPDL looks for a $u \in [0 : \lceil\sqrt{q}\rceil - 1]$ such that $\gamma^{-u\lceil\sqrt{q}\rceil} [\![\gamma^{y_i'}]\!]$ belongs to the hash map H. In such case, $H(\gamma^{-u\lceil\sqrt{q}\rceil} [\![\gamma^{y_i'}]\!]) = v$, which allows us to recover y_i' in $\mathcal{O}\big(\sqrt{q}\big)$ complexity.

Description of $\mathcal{A}_{\mathsf{PDL}}$. Now let us see how $\mathcal{A}_{\mathsf{PDL}}$ uses SubPDL to solve the PDL assumption. First, $\mathcal{A}_{\mathsf{PDL}}$ checks if $[\![x]\!] = [\![0]\!]$, and if true, it returns 0. In the following, we assume that $x \in \mathbb{F}_p^*$. The reduction finds a generator g of $\mathbb{F}_p^*$ as was described above. There exists $y \in [0 : p-1]$ such that $x = g^y \bmod p$.

For each $i \in [1:d]$, $\mathcal{A}_{\mathsf{PDL}}$ runs $\mathsf{SubPDL}([\![1, x]\!], g, p, p_i, e_i)$ to compute y_i. We obtain a

Table 2: Efficiencies of SubPDL on an input $([\![1,x]\!], g, p, q, e)$ and $\mathcal{A}_{\mathsf{PDL}}$ for $p-1=\prod_{i=1}^{d} p_i^{e_i}$. We only count the number of multiplication in $\mathbb{F}_p$ and scalar multiplications in $\mathbb{G}$, which dominate the computation, and the number of oracle calls to Exp.

	$\mathbb{F}_p$ mult.	$\mathbb{G}$ scalar mult.	Exp calls
SubPDL	$\approx 2\sqrt{q}$	$\approx 2\sqrt{q}$	e
$\mathcal{A}_{\mathsf{PDL}}$	$\approx 2\sum_{i=1}^{d}\sqrt{p_i}$	$\approx 2\sum_{i=1}^{d}\sqrt{p_i}$	$\sum_{i=1}^{d} e_i$

system of equations,

$$\begin{cases} y \equiv y_1 \pmod{p_1^{e_1}} \\ \vdots \\ y \equiv y_d \pmod{p_d^{e_d}} \end{cases}.$$

From the Chinese remainder theorem, $\mathcal{A}_{\mathsf{PDL}}$ can efficiently find $y \in [0:p-1]$ that satisfies the system of congruences. The complexity of this step is $\mathcal{O}\big(\log^2(p-1)\big)$ [MvOV01, Alg. 2.121]. Finally, it returns $x = g^y \bmod p$.

Remark 1. An alternative way to interpret the above algorithm is that if $p-1=\prod_{i=1}^{d} p_i^{e_i}$ is smooth and the assumption reveals $[\![x^{(p-1)/p_i^k}]\!]$ for all $k \in [1:e_i]$ and $i \in [1:d]$, then computing x is efficient. We are unaware of any real-world assumption that would provide such elements.

Efficiency. We provide a detailed computational efficiency overview of $\mathcal{A}_{\mathsf{PDL}}$ and SubPDL in Table 2. The computation bottleneck is $\mathcal{O}\Big(\sum_{i=1}^{d}\sqrt{p_i}\Big)$ scalar multiplication in $\mathbb{G}$, which why we need to require that $p-1$ is a somewhat smooth number. The bottleneck for the memory complexity is the size of the hash table. Since we need to store only one hash table at a time, the memory requirement is $\mathcal{O}\big(\sqrt{q}\cdot\log p\big)$ for $q=\max_{i=1}^{d} p_i$. The $\sum_{i=1}^{d} e_i$ oracle calls to Exp is also influential, which will later determine the tightness of the n-PDL $\Rightarrow$ n-DHE reduction.

3.2 n-PDL $\Rightarrow$ CDH

As a warm-up, let us show that n-PDL $\Rightarrow$ CDH for any $\mathsf{n} \geq 1$ using the framework developed above in Fig. 2. We assume that the CDH oracle always answers correctly, that is, $\mathsf{CDH}([\![a]\!], [\![b]\!]) = [\![ab]\!]$ for any $[\![a]\!], [\![b]\!] \in \mathbb{G}$. All that remains is implementing the Exp oracle with a CDH oracle.

First, we pick $m := \max_i((p-1)/p_i)$ and compute $[\![x^2]\!], [\![x^4]\!], \ldots, [\![x^{2^{\lfloor \log m \rfloor}}]\!]$. This will take $\lfloor \log m \rfloor$ calls to the CDH oracle. Next, computing $[\![x^{(p-1)/p_i^k}]\!]$ for some $k \in [1:e_i]$ also takes at most $\lfloor \log m \rfloor$ calls to CDH oracle since we can reuse the precomputed elements. In total, the reduction requires at most

$$\lfloor \log m \rfloor \cdot \left(1 + \sum_{i=1}^{d} e_i\right) \tag{3}$$

calls to the CDH oracle. We summarize the result below.

Theorem 1. *Let $\mathbb{G}$ be a group of prime order p and $p-1=\prod_{i=1}^{d} p_i^{e_i}$, where p_i are distinct prime factors. Let $\mathsf{n} \geq 1$. Suppose an algorithm $\mathcal{B}$ breaks the CDH assumption in $\mathbb{G}$ (with no error) in time T and with M bits of memory. Then, there exists a reduction $\mathcal{A}_{\mathsf{PDL}}$ that breaks n-PDL, by using $\mathcal{B}$ as a black-box, and it runs in time T', with M' bits of memory,*

Table 3: The table lists the number of 1s in the bit representation of $(p-1)/p_i^k$ and the number of CDH queries needed to compute $[\![x^{(p-1)/p_i^k}]\!]$ for BLS12-381 assuming that $[\![x^2, \ldots, x^{2^{253}}]\!]$ has been precomputed.

Element	$\frac{p-1}{p_1^{32}}$	$\frac{p-1}{p_2}$	$\frac{p-1}{p_3}$	$\frac{p-1}{p_4}$	$\frac{p-1}{p_5}$	$\frac{p-1}{p_6}$	$\frac{p-1}{p_7}$	$\frac{p-1}{p_8^2}$	$\frac{p-1}{p_8}$	$\frac{p-1}{p_9}$
#1s in bit rep.	133	111	108	109	104	103	106	92	105	114
#CDH calls	132	110	107	108	103	102	105	91	104	113

$\frac{p-1}{p_{10}}$	$\frac{p-1}{p_{11}}$	$\frac{p-1}{p_{12}^2}$	$\frac{p-1}{p_{12}}$	$\sum$
97	110	80	119	1491
96	109	79	118	1477

such that

$$T' \leq \lfloor \log m \rfloor \cdot \left(1 + \sum_{i=1}^{d} e_i\right) \cdot T + \mathcal{O}\left(\mathsf{sm} \cdot \sum_{i=1}^{d} \sqrt{p_i}\right) \ , \quad M' \leq \mathcal{O}(\sqrt{q} \cdot \log p) + M \ ,$$

where $m = \max_i((p-1)/p_i)$, sm *is the running time of a scalar multiplication, and* $q = \max_{i=1}^{d} p_i$.

Efficiency for BLS12-381. Let us now take a look at the efficiency of the n-PDL $\Rightarrow$ CDH reduction in the case of BLS12-381 group. The group order of BLS12-381 is

$$p = \quad \text{0x73eda753299d7d483339d80809a1d80553bda402fffe5bfeffffffff00000001} \ .$$

Importantly for us,

$$p - 1 = \quad 2^{32} \cdot 3 \cdot 11 \cdot 19 \cdot 10177 \cdot 125527 \cdot 859267 \cdot 906349^2 \cdot \\ 2508409 \cdot 2529403 \cdot 52437899 \cdot 254760293^2 \ ,$$

is 28bit smooth. Let us denote $p_1 = 2, p_2 = 3, \ldots, p_{12} = 254760293$. We show that many fewer CDH calls are needed for BLS12-381 than the Eq. (3) suggests. Let $m = {}^{(p-1)}/_2$, which has $b = \lfloor \log m \rfloor + 1 = 254$ bit representation. First, we precompute $[\![x^2, \ldots, x^{2^{253}}]\!]$, which requires 253 CDH calls. Observe that if a b bit number s has a binary representation $(s_0, \ldots, s_{b-1}) \in \{0,1\}^b$, then to compute $[\![x^s]\!] = [\![\prod_{i=0}^{b-1} (x^{2^i})^{s_i}]\!]$, we only have to perform $-1 + \sum_{i=0}^{b} s_i$ calls to the CDH oracle. That is one less than the number of 1s in the bit representation of s. We list the number of additional CDH calls (besides precomputation) needed to compute most elements of the form $[\![x^{(p-1)/p_i^k}]\!]$ in Table 3. The table does not contain $[\![x^{(p-1)/2^i}]\!]$ for $i = 1, 2, \ldots, 31$. These can be obtained more efficiently. Observe that if we know $[\![x^{(p-1)/2^i}]\!]$, then $\mathsf{CDH}([\![x^{(p-1)/2^i}]\!], [\![x^{(p-1)/2^i}]\!]) = [\![x^{(p-1)/2^{i-1}}]\!]$. Therefore, given $[\![x^{(p-1)/2^{32}}]\!]$, we can obtain the rest with only 31 CDH calls.

In total, we need only $253 + 1477 + 31 = 1761$ calls to the CDH oracle[6]. The best that [MS23] achieves is 12,308 CDH calls for the Anomalous curve. Thus, we obtain the least number of queries for dBM reduction known in the literature for any currently practical group, in addition to good memory complexity. The first target group, $\mathbb{G}_1$ in BLS12-381, has 381bit group elements in the compressed form and a memory requirement of roughly $381 \cdot \sqrt{p_{12}}$ bits $\approx$ 6.1 megabytes. The number of field multiplications is $2\sum_{i=1}^{d} \sqrt{p_i} < 60,000$, and the number of scalar multiplications in $\mathbb{G}_1$ is roughly the same.

[6]The number of oracle calls can be reduced further by optimal addition chains [Knu97, pp. 465-466].

3.3 n-PDL $\Rightarrow$ n-DHE

We show next that the hardness of the n-PDL assumption implies the hardness of n-DHE when $\mathsf{n} = \mathsf{poly}(\lambda)$ and $\mathsf{n} \geq 2$. The reduction gets $[\![1, x, \ldots, x^{\mathsf{n}}]\!]$ as an input and has to output x. Recall that the reduction has access to an oracle $\mathsf{O}_{\mathrm{dhe}}$ that takes as an input $[\![\mathsf{g}, y \cdot \mathsf{g}, \ldots, y^{\mathsf{n}} \cdot \mathsf{g}]\!]$ and outputs $[\![y^{\mathsf{n}+1} \cdot \mathsf{g}]\!]$, where g denotes the generator of the additive group $(\mathbb{F}_p, +)$. We rely on the framework developed in the previous section, which means that we only need to construct $\mathsf{Exp}(\cdot)$ oracle using $\mathsf{O}_{\mathrm{dhe}}(\cdot)$ oracle.

Linear Algorithm. The most straightforward approach for computing $[\![x^m]\!]$ is as follows. First, Exp runs $\mathsf{O}_{\mathrm{dhe}}([\![1, x, \ldots, x^{\mathsf{n}}]\!])$ to obtain $[\![x^{\mathsf{n}+1}]\!]$. Then it runs $\mathsf{O}_{\mathrm{dhe}}([\![x, x^2, \ldots, x^{\mathsf{n}+1}]\!])$ to obtain $[\![x^{\mathsf{n}+2}]\!]$. Here, the input has the correct structure $[\![\mathsf{g}, y \cdot \mathsf{g}, \ldots, y^{\mathsf{n}} \cdot \mathsf{g}]\!]$ when taking $\mathsf{g} = y = x$. Exp continues incrementing powers in a similar fashion by calling next $\mathsf{O}_{\mathrm{dhe}}([\![x^2, x^3, \ldots, x^{\mathsf{n}+2}]\!])$ to obtain $[\![x^{\mathsf{n}+3}]\!]$ and so on. To obtain $[\![x^m]\!]$ with this approach, it takes $m - n$ calls to the DHE oracle. Unfortunately, this algorithm requires an exponential number of calls in the security parameter λ when $m = (p-1)/p_i = \mathcal{O}(2^\lambda)$. Recall that p_i are the factors of the factorization of $p - 1$ as in $p - 1 = \prod_{i=1}^{d} p_i^{e_i}$.

However, we can slightly generalize this algorithm to compute sequences that form an arithmetic progression. Suppose we have a vector $L = [\![x^a, x^{a+d}, x^{a+2d}, \ldots, x^{a+\mathsf{n}d}]\!]$, where exponents form the beginning of an arithmetic series for some integers $a \geq 0$, $d > 0$. Running $\mathsf{O}_{\mathrm{dhe}}(L)$, we get the next element $[\![x^{a+(\mathsf{n}+1)d}]\!]$ in the series. By shifting the input of $\mathsf{O}_{\mathrm{dhe}}(\cdot)$ as before, we can obtain the first j elements of the series with $j - (\mathsf{n}+1)$ queries. This algorithm Arith is described in Fig. 3, and it will be useful in the efficient version of the exponentiation algorithm described next.

Fast Algorithm. In this section, we show how to compute $[\![x^m]\!]$ efficiently, given access to a n-DHE oracle and $[\![1, x^1, \ldots, x^{\mathsf{n}}]\!]$. The main idea is to, at each iteration i, keep a list of the first $2\mathsf{n}+2$ elements where the exponents form an arithmetic progression with the first term $a = m \bmod 2^i$ and the difference 2^i. This guarantees that m is an element of all these arithmetic progressions but not necessarily located in the first $2\mathsf{n}+2$ terms.

The algorithm works as follows:

1. Take the list $L = [\![1, x^1, \ldots, x^{\mathsf{n}}]\!]$ and duplicate the list with $\mathsf{n}+1$ calls to the DHE-oracle, to obtain $L_0 = [\![1, x^1, \ldots, x^{\mathsf{n}}, x^{\mathsf{n}+1}, \ldots, x^{2\mathsf{n}+1}]\!]$.
2. For each iteration $i = 1, \ldots, i_{\max}$, update the list with the arithmetic progression with $a = m \bmod 2^i$ and difference 2^i to obtain

$$L_i = [\![x^a, x^{a+2^i}, \ldots, x^{a+\mathsf{n}2^i}, x^{a+(\mathsf{n}+1)2^i}, \ldots, x^{a+(2\mathsf{n}+1)2^i}]\!].$$

   We will prove that the first $\mathsf{n}+1$ terms $[\![x^a, x^{a+2^i}, \ldots, x^{a+\mathsf{n}2^i}]\!]$ are inherited from the previous list L_{i-1} and thus can be used as input to the Arith algorithm (Fig. 3).

See Fig. 3 for a summarized explanation of the exponentiation algorithm.

Lemma 2. *The algorithm* $\mathsf{Exp}^{\mathsf{O}_{\mathrm{dhe}}(\cdot)}(m)$*, depicted in Fig. 3 is correct and requires* $\mathcal{O}((\mathsf{n}+1)\log(m/\mathsf{n})$ *calls to the* n-DHE *oracle* [7].

Proof. **Correctness)** It is not immediately obvious that Step 5 in Fig. 3 is possible. More precisely, let us denote $a_i := m \bmod 2^i$ and $a_{i-1} := m \bmod 2^{i-1}$. Suppose that the algorithm has already computed $L_{i-1} = [\![x^{a_{i-1}}, x^{a_{i-1}+2^{i-1}}, \ldots, x^{a_{i-1}+(2\mathsf{n}+1)2^{i-1}}]\!]$. Does it hold that L_{i-1} contains $[\![x^{a_i}, x^{a_i+2^i}, \ldots, x^{a_i+\mathsf{n}2^i}]\!]$ as claimed in the algorithm?

By modular division, we have $m = 2^i q_i + a_i = 2^{i-1} q_{i-1} + a_{i-1}$, where q_i, q_{i-1} are some positive integers and $0 \leq a_i < 2^i$ and $0 \leq a_{i-1} < 2^{i-1}$.

We examine two cases: $0 \leq a_i < 2^{i-1}$ and $2^{i-1} \leq a_i < 2^i$.

[7] A similar result was found independently by Lipmaa and Krips [LK25].

$\mathsf{Exp}^{\mathsf{O}_{\mathrm{dhe}}(\cdot)}(m, [\![1, x, \ldots, x^{\mathsf{n}}]\!])$

1 : **if** $m \leq \mathsf{n}$ **then return** $[\![x^m]\!]$;
2 : $L_{-1} \leftarrow [\![1, x, \ldots, x^{\mathsf{n}}]\!]$;
3 : $i \leftarrow 0; a \leftarrow 0$;
4 : **while** $m > a + (2\mathsf{n}+1)2^i$ **do**
5 : Pick $L \leftarrow [\![x^a, x^{a+2^i}, \ldots, x^{a+\mathsf{n}2^i}]\!]$ from L_{i-1};
6 : $L_i \leftarrow \mathsf{Arith}^{\mathsf{O}_{\mathrm{dhe}}(\cdot)}(2\mathsf{n}+1, L)$;
7 : // $L_i = [\![x^a, x^{a+2^i}, \ldots, x^{a+(\mathsf{n}+1)2^i}, \ldots, x^{a+(2\mathsf{n}+1)2^i}]\!]$
8 : $i \leftarrow i+1; a \leftarrow m \bmod 2^i$;
9 : **endwhile**
10 : **return** $[\![x^m]\!]$ from L_i;

$\mathsf{Arith}^{\mathsf{O}_{\mathrm{dhe}}(\cdot)}(j, [\![\mathsf{g}, y\mathsf{g}, \ldots, y^{\mathsf{n}}\mathsf{g}]\!])$

$i = 0$;
while $j \leq i + \mathsf{n} + 1$ **do**
$[\![y^{i+\mathsf{n}+1}\mathsf{g}]\!] \leftarrow \mathsf{O}_{\mathrm{dhe}}([\![y^i\mathsf{g}, y^{i+1}\mathsf{g}, \ldots, y^{i+\mathsf{n}}\mathsf{g}]\!])$;
$i \leftarrow i+1$;
return $[\![\mathsf{g}, y\mathsf{g} \ldots, y^j\mathsf{g}]\!]$;

Figure 3: The algorithm Exp to efficiently compute arbitrary exponentiation and the helper algorithm Arith.

First, suppose $0 \leq a_i < 2^{i-1}$. Since $2^{i-1}(2q_i) + a_i = 2^{i-1}q_{i-1} + a_{i-1}$ and reminders are unique, we have $a_i = a_{i-1}$. Therefore, any element of the form $x^{a_i + j2^i}$, for $j \in [0 : \mathsf{n}]$, is such that $x^{a_i+j2^i} = x^{a_{i-1}+(2j)2^{i-1}}$, and consequently, belongs to L_{i-1}.

Second, suppose $2^{i-1} \leq a_i < 2^i$. Then, we can write $a_i = 2^{i-1} + r$ for $r < 2^{i-1}$ since $a_i < 2^i$. We have

$$m = 2^i q_i + a_i = 2^i q_i + (2^{i-1} + r) = 2^{i-1}(2q_i + 1) + r = 2^{i-1}q_{i-1} + a_{i-1},$$

from the uniqueness of the Euclid's division lemma. Similarly, it follows that $r = a_{i-1}$. Hence, any element of the form $x^{a_i+j2^i}$, with $j \in [0 : \mathsf{n}]$, is such that $x^{a_i+j2^i} = x^{a_{i-1}+2^{i-1}+j2^i} = x^{a_{i-1}+2^{i-1}(1+2j)} \in L_{i-1}$.

Finally, let us also show that on Step 10, it is true that $[\![x^m]\!]$ belongs to the last list L_i. The **while** loop exits when $m \leq a + (2\mathsf{n}+1)2^i$, where $a = m \bmod 2^i$. Since $m = 2^i k + a$ for some integer $k \geq 0$ and by the condition above $k \leq 2\mathsf{n}+1$, it must be that $[\![x^m]\!]$ is one of the computed elements $[\![x^{a+j2^i}]\!]$ contained in L.

We conclude that the algorithm always terminates and outputs the correct value.

Efficiency) Each iteration requires $\mathsf{n}+1$ calls to the DHE-oracle. By definition and the fact that $m \equiv a \bmod 2^i$, for every $i \in [1 : i_{\max}]$, we know that m belongs to the arithmetic progression given by each list L_i. The remaining question is, how many iterations are necessary to find m? Indeed, we want

$$m \leq a + (2\mathsf{n}+1)2^{i_{\max}} \Rightarrow \tfrac{m-a}{2\mathsf{n}+1} \leq 2^{i_{\max}} \Rightarrow i_{\max} \geq \log\left(\tfrac{m-a}{2\mathsf{n}+1}\right) \approx \log\left(\tfrac{m}{\mathsf{n}}\right),$$

as we wanted to demonstrate. □

It follows that $\mathsf{Exp}^{\mathsf{O}_{\mathrm{dhe}}(\cdot)}(m)$ runs in polynomial time for any $\mathsf{n} = \mathsf{poly}(\lambda)$ and $m = \mathcal{O}(2^\lambda)$. Therefore, n-PDL $\Rightarrow$ n-DHE.

Discussion. It is natural to ask if DHE (resp. CDH), being computationally equivalent to PDL (resp. DL), has a positive or a negative implication for groups like BLS12-381. We find it impossible to judge at the current point, but understanding the relationship between different assumptions is important. The fact that there are practical groups with highly smooth $p-1$ is curious, but we are unaware that PDL or DL would be any faster to break in these groups. In the case of factorization, Pollard's $p-1$ algorithm [WJ13, pp. 138-140] takes advantage of the fact that a factor p has a smooth $p-1$. This leads to the notion of strong primes in cryptography. Maurer [Mau94] cautions against constructing groups with a smooth $p-1$ for the sole reason of having a fast reduction between DL and CDH. He reasons that it might not be worth it and there could be some unknown negative

consequences for the smooth $p-1$. In the case of pairing groups, the smooth $p-1$ leads to faster SNARKs, which has a more significant benefit.

4 Impossibility Results

4.1 DHE With Fixed Base Generator

In Section 3.3, we allowed the reduction to change the generator $[\![1]\!]$. The reduction could call $\mathsf{O}_{\mathrm{dhe}}$ on any input $[\![\mathsf{g}, a\mathsf{g}, \ldots, a^{\mathsf{n}}\mathsf{g}]\!]$ as long as $[\![\mathsf{g}]\!]$ is the generator of the group $\mathbb{G}$. Imagine a more restrictive oracle that requires $[\![\mathsf{g}]\!] = [\![1]\!]$, i.e., the generator is fixed base. We call this assumption $\mathsf{DHE}_{\mathrm{fb}}$. We show that there is no efficient generic reduction R, which would show that the hardness of n-PDL implies hardness of m-$\mathsf{DHE}_{\mathrm{fb}}$, for any m and $n < 2m$. Observe that dBM reduction (either for CDH or DHE) is generic. It treats the group as a black-box and performs only generic group operations with the group elements, namely, additions, equality tests, and CDH (or DHE) queries. Therefore, our result implies that fundamentally different reduction techniques are needed when one does not allow a variable base generator.

We start formalizing a security game for such a generic reduction algorithm. Intuitively, we would like to use the generic Add and Enc oracles from Fig. 1. In such case, the game samples a random challenge $x \leftarrow\!\!\$\ \mathbb{F}_p$ and the reduction R gets as an input encodings $\mathsf{Enc}(1)$, $\mathsf{Enc}(x)$, …, $\mathsf{Enc}(x^{\mathsf{n}})$. The reduction also gets an oracle access to Add and $\mathsf{DHE}_{\mathrm{fb}}$. The $\mathsf{DHE}_{\mathrm{fb}}$ oracle takes as an input encodings $\xi_0, \ldots, \xi_m$, checks that they match encodings of $\mathsf{Enc}(1), \mathsf{Enc}(w), \ldots, \mathsf{Enc}(w^m)$ for some $w \in \mathbb{F}_p$, and returns $\mathsf{Enc}(w^{m+1})$. All the encodings that the reduction receives are stored in a vector $\vec{\sigma}$ and the corresponding field elements in a vector $\vec{x}$. The reduction wins by guessing the challenge x.

Since we require $\xi_0 = \mathsf{Enc}(1)$, it is not possible for R to change the generator. The oracle tests if the input is well-formed, but it does not have to be distributed equivalently to an actual m-$\mathsf{DHE}_{\mathrm{fb}}$ instance, i.e., we do not require that w is uniformly random. This relaxation strengthens our subsequent impossibility result. Note that our result does not rule out reductions, which give ill-formed inputs to the oracle. This is because the oracle is usually an efficient algorithm in reduction proofs. Thus, some reduction could exist that provides the oracle inputs computationally indistinguishable from the honest ones.

The final game $\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda)$ slightly deviates from the above description. Since each element in $\vec{x}$ is an addition or exponentiation of the prior elements, then for each x_i, there exists a natural polynomial $f_i(X)$ such that $f_i(x) = x_i$. For the proof of Theorem 2, it is convenient to store these polynomials in a vector $\vec{f}$. To accommodate that, we define $\overline{\mathsf{Enc}}$, which takes the respective polynomial $f_i(X)$ as an input and stores it in the vector $\vec{f}$, but otherwise behaves identically to $\mathsf{Enc}(f_i(x))$. We also use a modified $\overline{\mathsf{Add}}$ oracle that computes the addition polynomial $f_i(X)$ and returns $\overline{\mathsf{Enc}}(f_i(X))$. We refer the reader to Fig. 4 for the complete description of the game.

Theorem 2. *Let $m \geq 2$ and consider a group of prime order p. For any τ-time reduction R and $n < 2m$,*

$$\Pr\left[\mathsf{Game}^{dhe\text{-}ggm}_{\mathsf{n},m,R}(1^\lambda) = 1\right] = \mathcal{O}\Big(\tfrac{(\mathsf{n}^3+\mathsf{n}^2\tau+\mathsf{n}\tau^2)\cdot m^2}{p}\Big) \ .$$

Therefore, for any $m, \tau \in \mathsf{poly}(\lambda)$, for any $n < 2m$, and $p \in \omega(\mathsf{poly}(\lambda))$, we have n-PDL $\overset{\text{gen}}{\not\Longrightarrow}$ m-$\mathsf{DHE}_{\mathrm{fb}}$.

Proof. Let R be a τ-time reduction. We present a hybrid game $\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda)$ in Fig. 4, which differs in the following way from the $\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda)$:

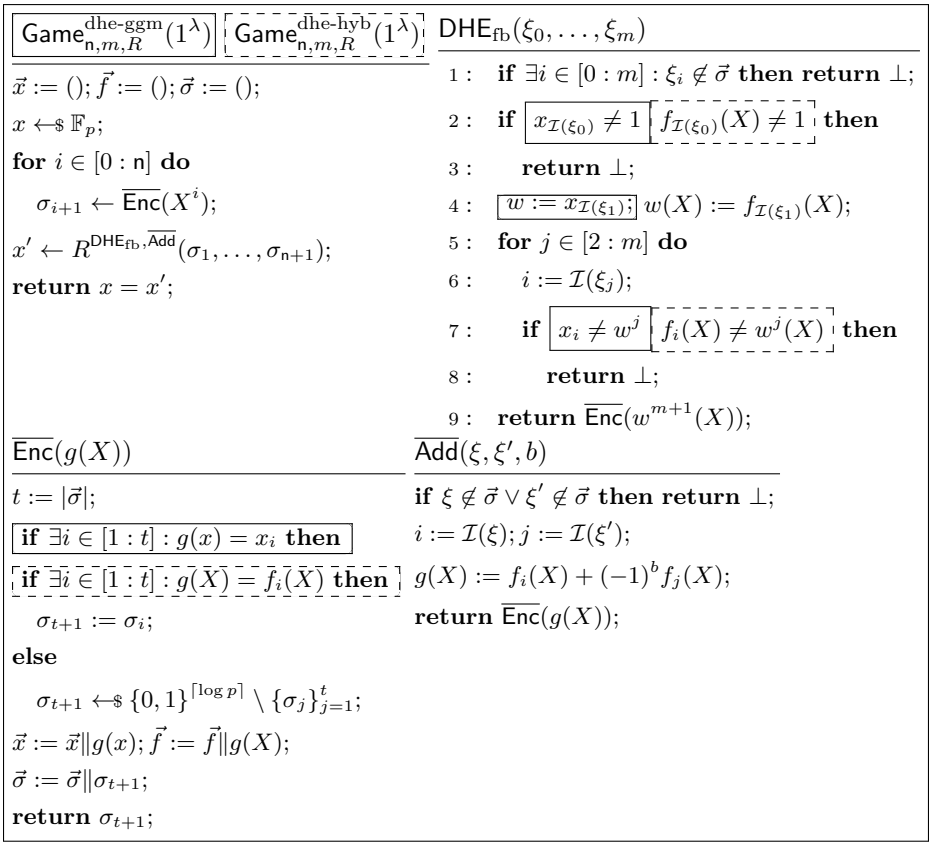

Figure 4: The GGM game $\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda)$ for the PDL/$\mathsf{DHE}_{\mathrm{fb}}$ reduction R where the generator is fixed base and its hybrid game $\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda)$ from Theorem 2. The steps present only in the first game are denoted by $\boxed{box}$, and the steps present only in the hybrid game are denoted by $\lceil box \rceil$. The algorithm $\mathcal{I}$ is described in Fig. 1.

- The encoding function $\overline{\mathsf{Enc}}$ encodes the polynomial $f_i(X)$ instead of the corresponding field element x_i. If $\overline{\mathsf{Enc}}$ gets $g \notin \vec{f}$ as an input, it assigns to g a new randomly sampled encoding. Otherwise, $\overline{\mathsf{Enc}}$ returns the preexisting encoding. $\overline{\mathsf{Add}}$ is identical in both games except for the slight difference in the $\overline{\mathsf{Enc}}$ oracle, which $\overline{\mathsf{Add}}$ also calls.
- $\mathsf{DHE}_{\mathrm{fb}}$ tests the well-formedness of the input on the polynomials corresponding to the encodings instead of the field elements. See Lines 2 and 7 in Fig. 4.

The inputs R receives in the hybrid game are chosen independently of the challenge x. Hence, the probability that R guesses x is bounded by $1/p$, $\Pr[\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda) = 1] \leq 1/p$.

We say that the event bad happens when the final state of $\vec{f}$ satisfies,

$$\exists g, h \in \vec{f}, k \in [1 : m] : g(X) \neq h^k(X) \wedge g(x) = h^k(x) \ ,$$

where x is the challenge. We claim that

$$\Pr[\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda) = 1 \wedge \neg\mathsf{bad}] \leq \Pr[\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda) = 1 \wedge \neg\mathsf{bad}] \ . \tag{4}$$

Let us fix some random coins $r = r_{\overline{\mathsf{Enc}}} \| r_R$ for $\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda)$, where $r_{\overline{\mathsf{Enc}}}$ are the coins used for $\overline{\mathsf{Enc}}$ and r_R are the coins used by the reduction R. We show that if the event

$\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda) = 1 \wedge \neg\mathsf{bad}$ happens on random coins r, then on the same random coins r also the event $\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda) = 1 \wedge \neg\mathsf{bad}$ happens. More precisely, we will show that given that $\neg\mathsf{bad}$ happens, the input to R will be the same in both games, and consequently, the output of R. This happens since the output of R is uniquely determined by its inputs and the random coins r_R. Therefore, Eq. (4) must be satisfied.

- Initially R gets $\overline{\mathsf{Enc}}(1), \overline{\mathsf{Enc}}(X), \ldots, \overline{\mathsf{Enc}}(X^{\mathsf{n}})$ as an input in the original game. Since $X^i, X^j \in \vec{f}$ for any $0 \le i < j \le \mathsf{n}$, the event $\neg\mathsf{bad}$ implies that $x^i \neq x^j$. Therefore, $1, x, \ldots, x^{\mathsf{n}}$ get assigned random and independently chosen encodings in the initial game. In the hybrid game, 1, X, $\ldots$, X^{n} are distinct polynomials, and thus they get assigned the same random and independently chosen encodings.

- Assume inductively that inputs to R have been identical in $\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda)$ and $\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda)$ up until some $\overline{\mathsf{Add}}$ query (ξ, ξ', b). In such case, also the states of $\vec{x}$, $\vec{f}$, and $\vec{\sigma}$ are identical in both games. Therefore, $\xi \notin \vec{\sigma} \vee \xi' \notin \vec{\sigma}$ in one of the games implies it is also true in the other. Thus, in both games, the response will be $\perp$. If the input is well-formed, the response will be $\overline{\mathsf{Enc}}(g(X))$ in both games. By the event $\neg\mathsf{bad}$, for all $h \in \vec{f}$, we have $g(x) \neq h(x)$ if $g \neq h$. Therefore, $g(x) \in \vec{x}$ if an only if $g(X) \in \vec{f}$. That means $\overline{\mathsf{Enc}}(g(X))$ will give the same response in both games.

- Assume inductively that inputs to R have been identical in $\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda)$ and $\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda)$ up until some DHE_{fb} query $(\xi_0, \ldots, \xi_m)$. If DHE_{fb} terminates on Step 1 or Step 9, the analysis is identical to Add from above. Below we consider termination on Steps 2 and 7. Recall from Fig. 1, for a vector of encodings $\vec{\sigma}$, we denote $\mathcal{I}(\xi) = \min\{k \in [1 : |\vec{\sigma}|] : \xi = \sigma_k\}$.

  - Assume $f_{\mathcal{I}(\xi_0)}(X) \neq 1$ holds on the Step 2 of $\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda)$. Since $1 \in \vec{f}$, the event $\neg\mathsf{bad}$ implies that $f_{\mathcal{I}(\xi_0)}(x) = x_{\mathcal{I}(\xi_0)} \neq 1$. Vice-versa, if $x_{\mathcal{I}(\xi_0)} \neq 1$, then $f_{\mathcal{I}(\xi_0)}(X)$ is not a constant polynomial 1. Therefore, the condition in Line 2 is satisfied in $\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda)$ if and only if it is satisfied in $\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda)$. If one of the games outputs $\perp$ on Step 2, so does the other.
  - Suppose the condition $f_i(X) \neq w^j(X)$ is satisfied on Step 7 of $\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda)$. In the $\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda)$, we have $f_i(x) = x_i$ and $w^j(x) = w^j$. According to the event $\neg\mathsf{bad}$, $f_i(X) \neq w^j(X)$ implies that $f_i(x) \neq w^j(x)$. Therefore, $f_i(X) \neq w^j(X)$ holds if and only if $f_i(x) \neq w^j(x)$. It follows that if DHE_{fb} outputs $\perp$ at Step 7 in $\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda)$, then the same happens in $\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda)$.

Now, let us analyze the probability of the event bad. Consider the polynomials in the vector $\vec{f}$ after R has outputted x' in the hybrid game. We define $\deg(\vec{f}) := \max_{g \in \vec{f}} \deg(g)$ and show that $\deg(\vec{f}) \le \mathsf{n} + \tau$.

Before any queries, we have $\vec{f} = (1, X, \ldots, X^{\mathsf{n}})$. Suppose $(\xi_0, \ldots, \xi_m)$ is the first DHE_{fb} query that R makes, which does not result in a $\perp$ output. We can conclude that the encodings $(\xi_0, \ldots, \xi_m)$ correspond to some polynomials in $\vec{f}$ of the form $1, w(X), w^2(X), \ldots, w^m(X)$. Since $\overline{\mathsf{Add}}$ queries do not change $\deg(\vec{f})$, it must be that $\deg(\vec{f}) = \mathsf{n}$ before the query. Thus, $\deg(w^m) = m \deg(w) \le \mathsf{n}$ and it follows that

$$\deg(w) \le \mathsf{n}/m \ . \tag{5}$$

If $\mathsf{n} < m$, then Eq. (5) implies that $\deg(w) \le 0$. Now we obtain a bound on the output polynomial $\deg(w^{m+1}) = (m+1) \cdot \deg(w) \le 0$ and after the first DHE_{fb} query $\deg(\vec{f}) = \mathsf{n}$ just as before the query. If $m \le \mathsf{n} < 2m$ then $n/m < 2$, and Eq. (5) implies that $\deg(w) \le 1$.

Therefore, $\deg(w^{m+1}) \leq \deg(w^{\mathsf{n}+1}) \leq \mathsf{n}+1$. In either case ($\mathsf{n} = m$ or $\mathsf{n} < m$), we can conclude that after the first query $\deg(\vec{f}) \leq \mathsf{n}+1$.

Consider now the second query to the $\mathsf{DHE}_{\mathrm{fb}}$ oracle. Let it correspond to some polynomials $1, u(X), u^2(X), \ldots, u^m(X) \in \vec{f}$. Suppose $\deg(u) \geq 2$. Then we obtain a contradiction since for $\deg(u^m) = m \cdot \deg(u) \geq 2m$, which is impossible since $\vec{f}$ contains polynomials of degree at most $\mathsf{n}+1 \leq m+1 < 2m$. Thus, it must be that $\deg(u) \leq 1$. However, the output polynomial will again be at most degree $\mathsf{n}+1$. We conclude that after R has finished its execution, $\deg(\vec{f}) \leq \mathsf{n}+1$.

Since R makes at most τ operations, $\vec{f}$ will contain at most $\mathsf{n}+1+\tau$ distinct polynomials, each of degree at most $\mathsf{n}+1$. Let $f_i, f_j \in \vec{f}$ and $k \in [1:m]$ such that $f_i(X) \neq f_j^k(X)$. The probability that $f_i(x) = f_j^k(x)$ is bounded by $(\mathsf{n}+1)k/p$ since $f_j^k(X) - f_i(X)$ is a non-zero polynomial with at most $k(\mathsf{n}+1)$ roots. The probability that for any $i, j \in \{1, \mathsf{n}+1+\tau\}$, $k \in [1:m]$, $f_i(x) = f_j^k(x)$ is

$$\Pr[\mathsf{bad}] \leq \sum_{k=1}^{m} \frac{(\mathsf{n}+1)k}{p} \cdot (\mathsf{n}+1+\tau)^2 = \frac{(\mathsf{n}+1)(\mathsf{n}+1+\tau)^2(m+1)m}{2p} \ .$$

It follows that

$$\begin{aligned}\Pr[\mathsf{Game}^{\text{dhe-ggm}}_{\mathsf{n},m,R}(1^\lambda) = 1] &\leq \Pr[\mathsf{bad}] + \Pr[\mathsf{Game}^{\text{dhe-hyb}}_{\mathsf{n},m,R}(1^\lambda) = 1 \wedge \neg\mathsf{bad}] \\ &\leq \Pr[\mathsf{bad}] + 1/p = \mathcal{O}\Big(\tfrac{(\mathsf{n}^3+\mathsf{n}^2\tau+n\tau^2)m^2}{p}\Big). \qquad \square\end{aligned}$$

4.2 n-PDL $\overset{gen}{\nRightarrow}$ m-GDHE

Bauer, Fuchsbauer, and Loss [BFL20] showed that even very general Uber assumptions reduce to the PDL assumption in the Algebraic Group Model (see Appendix B.2 for the definition). We show that dBM techniques are likely to be insufficient to show the same results in the standard model. We recall the n-Generalized Diffie-Hellman Exponent (n-GDHE) assumption, which is a generalization of the n-DHE assumption.

Definition 4 (n-GDHE)**.** We say that the n-Generalized Diffie-Hellman Exponent assumption is secure respect to Ggen if for any PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\text{gdhe}}_{\mathsf{Ggen},\mathcal{A},\mathsf{n}}(\lambda) := \Pr\left[\ [\![x^{\mathsf{n}}]\!] = \mathcal{A}\left(\begin{array}{c}\mathsf{p}, [\![1, x, \ldots, x^{\mathsf{n}-1}]\!], \\ [\![x^{\mathsf{n}+1}, \ldots, x^{2\mathsf{n}}]\!]\end{array}\right) \ \middle|\ \begin{array}{c}\mathsf{p} \leftarrow \mathsf{Ggen}(1^\lambda), \\ x \leftarrow\!\$\ \mathbb{F}_p\end{array}\ \right] \approx_\lambda 0.$$

It is easy to see that n-GDHE $\Rightarrow$ $(\mathsf{n}-1)$-DHE. Suppose that there exists an efficient algorithm $\mathcal{A}$ that breaks the $(\mathsf{n}-1)$-DHE assumption. Then, a n-GDHE adversary that gets $[\![1, x, \ldots, x^{\mathsf{n}-1}, x^{\mathsf{n}-1}, \ldots, x^{2\mathsf{n}}]\!]$ as an input can return the output of $\mathcal{A}([\![1, x, \ldots, x^{\mathsf{n}-1}]\!])$ to break n-GDHE assumption. Therefore also n-GDHE $\Rightarrow$ $(\mathsf{n}-1)$-PDL. The opposite direction (with slightly different parameters) is known to be true in the algebraic group model. We sketch the main idea below.

Theorem 3. $(2\mathsf{n})$-PDL $\overset{\text{alg}}{\Longrightarrow}$ n-GDHE.

Sketch. Let $\mathcal{A}$ be an algebraic adversary that breaks the n-GDHE assumption with non-negligible probability. We construct an efficient adversary $\mathcal{B}$ that breaks $(2\mathsf{n})$-PDL assumption with non-negligible probability. $\mathcal{B}$ receives $[\![1, x, \ldots, x^{2\mathsf{n}}]\!]$ as an input and it runs $\mathcal{A}$ on an input $[\![1, x, \ldots, x^{\mathsf{n}-1}, x^{\mathsf{n}-1}, \ldots, x^{2\mathsf{n}}]\!]$ (i.e., without the element $[\![x^{\mathsf{n}}]\!]$). A successful $\mathcal{A}$ will output $[\![x^{\mathsf{n}}]\!]$ with some integers $a_0, \ldots, a_{\mathsf{n}-1}, a_{\mathsf{n}-1}, \ldots, a_{2\mathsf{n}}$ such that $[\![x^{\mathsf{n}}]\!] = \sum_{i=0}^{\mathsf{n}-1} a_i [\![x^i]\!] + \sum_{i=\mathsf{n}+1}^{2\mathsf{n}} a_i [\![x^i]\!]$.

Observe that $f(X) := X^{\mathsf{n}} - \sum_{i=0}^{\mathsf{n}-1} a_i X^i - \sum_{i=\mathsf{n}+1}^{2\mathsf{n}} a_i X^i$ is a non-zero polynomial with $f(x) = 0$. Therefore, $\mathcal{B}$ can find the roots of $f(X)$, find the one that matches the discrete logarithm of $[\![x]\!]$, and output x. $\square$

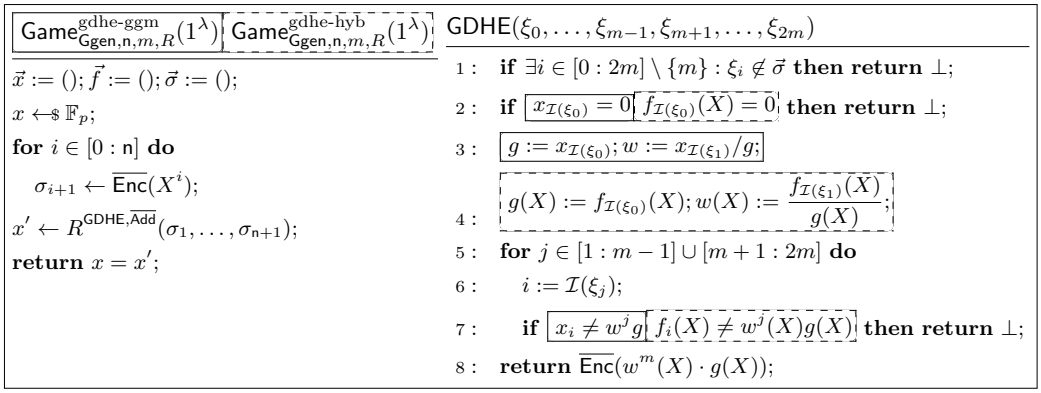

Figure 5: The GGM game $\mathsf{Game}^{\text{gdhe-ggm}}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda)$ for the PDL/GDHE reduction R and its hybrid game $\mathsf{Game}^{\text{gdhe-hyb}}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda)$ from Theorem 4. The steps present only in the first game are denoted by $\boxed{box}$, and the steps present only in the hybrid game are denoted by $\boxed{box}$. $\mathcal{I}$ is described in Fig. 1 and $\overline{\mathsf{Add}}$ and $\overline{\mathsf{Enc}}$ in Fig. 4.

We find it somewhat surprising that n-PDL $\overset{\text{gen}}{\not\Rightarrow}$ m-GDHE as we show next. However, this result follows the intuition from the previous section. An oracle that solves m-GDHE instances cannot compute $[\![x^t]\!]$ for $t \gg m$. We present the reduction game $\mathsf{Game}^{\text{gdhe-ggm}}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda)$ in Fig. 5, analogous to the one in Fig. 4. For the sake of completeness, we still provide a formal proof the following result in Appendix D.

Theorem 4. *Consider a group of prime order p. For any τ-time reduction R and $\mathsf{n} \geq 1$, $m \geq 2$,*

$$\Pr\left[\mathsf{Game}^{gdhe\text{-}ggm}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda) = 1\right] = \mathcal{O}\left(\frac{m^2\mathsf{n}(\mathsf{n}^3+\mathsf{n}^2\tau+\mathsf{n}\tau^2+\tau^3)}{p}\right) .$$

Therefore, for any $\mathsf{n}, m, \tau \in \mathsf{poly}(\lambda)$ such that $\mathsf{n} \geq 1$, $m \geq 2$ and $p \in \omega(\mathsf{poly}(\lambda))$, we have n-PDL $\overset{\text{gen}}{\not\Rightarrow}$ m-GDHE.

5 From Adversaries to Perfect Oracles

5.1 Revisiting Shoup's Perfect CDH Oracle

Let β, ε be arbitrary parameters such that $\beta > 0, \varepsilon < 1$. [Sho97] defines an efficient adversary that wins the CDH game with probability at least $1 - \beta$, given oracle access to an adversary ε-CDH that wins the game with probability at least ε.

Theorem 5 ([Sho97])**.** *Given an oracle ε-CDH that wins the CDH game with probability ε, we can construct an algorithm that solves CDH with the following properties. For each $\beta \in (0, 1)$, it holds that*

1. *the algorithm solves CDH with probability at least $1 - \beta$,*
2. *it makes $\mathcal{O}(1/\varepsilon)\log(1/\beta)$ queries to the imperfect ε-CDH oracle,*
3. *it runs in time $\mathcal{O}(1/\varepsilon)\log(1/\beta)$.*

For simplicity, we consider group exponentiations as operations that are constant in the security parameter, despite they are performed in time $\mathcal{O}(\log p)$. Note that if ε is greater than the inverse of a polynomial, in the security parameter (i.e. the CDH assumption would not hold), then the algorithm defined according to the previous theorem is a PPT.

Results in [Sho97] only argue on the asymptotic behavior of the defined adversary. Moreover, it assumes that ε is known. This suffices for the purpose of using the amplification

technique for a reduction from CDH to DL. In fact, since we only need to show that for every CDH adversary, there exists a DL one, the latter can be defined as having the right ε hard-coded. However, the reduction defined in this way is only of theoretical interest.

In this section, we revisit [Sho97] result. First, we show how to define the algorithm that solves CDH with arbitrarily high probability, giving a concrete bound for any concrete choice of the parameters. In particular, for each $\beta \in (0,1)$, we show how to define the most efficient adversary that solves CDH at least with probability $1-\beta$, having black-box access to an arbitrary CDH adversary ε-CDH. As a sub-task, the defined amplified CDH adversary needs to estimate ε, having black-box access to ε-CDH. This sub-task is performed independently from the input. Thus, the reduction from CDH to DL can estimate the advantage of the given CDH adversary only once at the beginning.

Corollary 1. *Let $\beta \in (0,1)$ be an arbitrary constant. Suppose the* CDH *assumption does not hold. Thus, there exists a PPT $\mathcal{B}$ and a polynomial $s(X)$, such that, there exists an integer λ_0, such that $\mathsf{Adv}^{\mathsf{cdh}}_{\mathcal{B}}(\lambda) \geq 1/s(\lambda)$, for each $\lambda \geq \lambda_0$. Then, there exists an expected PPT* CDH *adversary $\mathcal{A}$ with the following properties.*

1. $\Pr\left[\ u \neq x, y \ \middle|\ x, y \leftarrow\!\!\$\ \mathbb{F}_p; [\![u]\!] \leftarrow \mathcal{A}([\![1, x, y]\!], \beta);\ \right] \geq 1 - \beta - 2^{-\lambda}$.
2. *$\mathcal{A}$ makes $1 + 50(\lambda - 1) + \mathcal{O}(1/\varepsilon)\log(1/\beta)$ calls, on average, to the adversary $\mathcal{B}$. Only the last $\mathcal{O}(1/\varepsilon)\log(1/\beta)$ calls depend on the input $[\![x, y]\!]$.*

The proof can be found in Appendix E.1. A concrete algorithm was defined in [MW96], which is asymptotically less efficient than the one in [Sho97],. The algorithm in this section is as practical as the one from [MW96] and asymptotically as efficient as the one in [Sho97].

5.2 A Perfect DHE Oracle

The results from Section 5.1 can be adapted to define a reliable n-DHE oracle. A crucial feature is that, as in the case of CDH, it is possible to randomize a DHE challenge. Let $[\![\{x^i\}_{i=0}^{\mathsf{n}}]\!]$, and let $a, b \in \mathbb{F}_p$. We can compute a valid DHE challenge $[\![\{(ax+b)^i\}_{i=0}^{\mathsf{n}}]\!]$, by evaluating the polynomial $(aX+b)^i$ in x.

Moreover, we can recover the answer $[\![x^{\mathsf{n}+1}]\!]$ given the correct answer $[\![u]\!]$, where $u = (ax+b)^{\mathsf{n}+1}$, and field elements a, b. Since $(aX+b)^{\mathsf{n}+1} = a^{\mathsf{n}+1}X^{\mathsf{n}+1} + f(X, a, b)$, where $f(X, A, B)$ is a polynomial of degree in X up to n, we get $[\![x^{\mathsf{n}+1}]\!] = ([\![u]\!] - [\![f(x, a, b)]\!])/a^{\mathsf{n}+1}$.

These properties are needed to get answers from the n-DHE oracle that are pairwise independent for the challenge $[\![x^{\mathsf{n}+1}]\!]$.

Corollary 2. *Let $\beta \in (0,1)$ be an arbitrary constant and n be an integer value that is bounded by a polynomial in the security parameter. Suppose the* n-DHE *assumption does not hold. Thus, there exists a PPT $\mathcal{B}$ and a polynomial $s(X)$, such that, there exists an integer λ_0, such that $\mathsf{Adv}^{\mathsf{n,dhe}}_{\mathcal{B}}(\lambda) \geq 1/s(\lambda)$, for each $\lambda \geq \lambda_0$. Then, there exists an expected PPT* DHE *adversary $\mathcal{A}$ with the following properties.*

1. $\Pr\left[\ u \neq x^{\mathsf{n}+1} \ \middle|\ x \leftarrow\!\!\$\ \mathbb{F}_p; [\![u]\!] \leftarrow \mathcal{A}([\![\{x^i\}_{i=0}^{\mathsf{n}}]\!], \beta);\ \right] \geq 1 - \beta - 2^{-\lambda}$.
2. *$\mathcal{A}$ makes $1 + 50(\lambda - 1) + \mathcal{O}(1/\varepsilon)\log(1/\beta)$ calls, on average, to the adversary $\mathcal{B}$. Only the last $\mathcal{O}(1/\varepsilon)\log(1/\beta)$ calls depend on the input $[\![x, y]\!]$.*

The proof is similar to the one of Corollary 1 and outlined in Appendix E.2.

6 Acknowledgement

The authors sincerely thank anonymous reviewers for improving an earlier version of this manuscript, particularly for their suggestion that significantly optimized algorithm $\mathsf{Exp}(\cdot)$. Bollauf and Siim were co-funded by the European Union and Estonian Research Council via grant PRG2531 and project TEM-TA119.

References

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, Berlin, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_14.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Berlin, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_4.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Berlin, Heidelberg, May 2005. doi:10.1007/11426639_26.

[Ben05] K. Bentahar. The equivalence between the DHP and DLP for elliptic curves used in practical applications, revisited. In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 376–391. Springer, Berlin, Heidelberg, December 2005. doi:10.1007/11586821_25.

[BFL20] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 121–151. Springer, Cham, August 2020. doi:10.1007/978-3-030-56880-1_5.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. URL: https://eprint.iacr.org/2017/1050.

[BLS03] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 257–267. Springer, Berlin, Heidelberg, September 2003. doi:10.1007/3-540-36413-7_19.

[BN06] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography*, pages 319–331, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Boy08] Xavier Boyen. The uber-assumption family. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008*, pages 39–56, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Cam13] Philippe Camacho. Fair exchange of short signatures without trusted third party. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *LNCS*, pages 34–49. Springer, Berlin, Heidelberg, February / March 2013. doi:10.1007/978-3-642-36095-4_3.

[CF13] Dario Catalano and Dario Fiore. Practical homomorphic MACs for arithmetic circuits. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 336–352. Springer, Berlin, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_21.

[Che06] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11. Springer, Berlin, Heidelberg, May / June 2006. doi:10.1007/11761679_1.

[Con24] Consensys. Gnark crypto. https://github.com/Consensys/gnark-crypto, 2024.

[den90] Bert den Boer. Diffie-Hellman is as strong as discrete log for certain primes (rump session). In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 530–539. Springer, New York, August 1990. doi:10.1007/0-387-34799-2_38.

[Den02] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 100–109. Springer, Berlin, Heidelberg, December 2002. doi:10.1007/3-540-36178-2_6.

[DGNW20] Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. Pixel: Multi-signatures for consensus. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 2093–2110. USENIX Association, August 2020.

[Eth24] Ethereum. Bls signatures. https://github.com/ethereum/consensus-specs/blob/v1.0.0/specs/phase0/beacon-chain.md#bls-signatures, 2024.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. doi:10.1007/978-3-319-96881-0_2.

[Gen04] Rosario Gennaro. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 220–236. Springer, Berlin, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8_14.

[GG17] Essam Ghadafi and Jens Groth. Towards a classification of non-interactive computational assumptions in cyclic groups. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 66–96. Springer, Cham, December 2017. doi:10.1007/978-3-319-70697-9_3.

[JM24] Joseph Jaeger and Deep Inder Mohan. Generic and algebraic computation models: When AGM proofs transfer to the GGM. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part V*, volume 14924 of *LNCS*, pages 14–45. Springer, Cham, August 2024. doi:10.1007/978-3-031-68388-6_2.

[Knu97] Donald E. Knuth. *The Art of Computer Programming*, volume 2: Seminumerical Algorithms. Addison-Wesley, 3rd edition, 1997.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_11.

[Laj22] Lóczi Lajos. Guaranteed- and high-precision evaluation of the lambert w function. *Applied Mathematics and Computation*, 433:127406, 2022. URL: https://www.sciencedirect.com/science/article/pii/S0096300322004805, doi:10.1016/j.amc.2022.127406.

[LK25] Helger Lipmaa and Toomas Krips. Private communication, 2025.

[LPS23] Helger Lipmaa, Roberto Parisella, and Janno Siim. Algebraic group model with oblivious sampling. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 363–392. Springer, Cham, November / December 2023. doi:10.1007/978-3-031-48624-1_14.

[LPS24] Helger Lipmaa, Roberto Parisella, and Janno Siim. Constant-size zk-SNARKs in ROM from falsifiable assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 34–64. Springer, Cham, May 2024. doi:10.1007/978-3-031-58751-1_2.

[Mau94] Ueli M. Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete algorithms. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 271–281. Springer, Berlin, Heidelberg, August 1994. doi:10.1007/3-540-48658-5_26.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, December 2005. doi:10.1007/11586821_1.

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Berlin, Heidelberg, December 2016. doi:10.1007/978-3-662-53887-6_27.

[MS23] Alexander May and Carl Richard Theodor Schneider. Dlog is practically as hard (or easy) as DH - solving dlogs via DH oracles on EC standards. *IACR TCHES*, 2023(4):146–166, 2023. doi:10.46586/tches.v2023.i4.146-166.

[MSV04] A. Muzereau, N. P. Smart, and F. Vercauteren. The equivalence between the dhp and dlp for elliptic curves used in practical applications. *LMS Journal of Computation and Mathematics*, 7:50–72, 2004. doi:10.1112/S1461157000001042.

[MvOV01] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[MW96] Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 268–282. Springer, Berlin, Heidelberg, August 1996. doi:10.1007/3-540-68697-5_21.

[MW99] Ueli Maurer and Stefan Wolf. The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, 28(5):1689–1721, 4 1999.

[Net24] Chia Network. Bls signatures. https://github.com/Chia-Network/bls-signatures, 2024.

[PH78] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms overgf(p)and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1):106–110, 1978. doi:10.1109/TIT.1978.1055817.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013. doi:10.1109/SP.2013.47.

[PSNB11] Geovandro C.C.F. Pereira, Marcos A. Simplício, Michael Naehrig, and Paulo S.L.M. Barreto. A family of implementation-friendly bn elliptic curves. *Journal of Systems and Software*, 84(8):1319–1326, 2011. URL: https://www.sciencedirect.com/science/article/pii/S0164121211000914, doi:10.1016/j.jss.2011.03.083.

[Rot22] Lior Rotem. Revisiting the Uber Assumption in the Algebraic Group Model: Fine-Grained Bounds in Hidden-Order Groups and Improved Reductions in Bilinear Groups. In Dana Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography (ITC 2022)*, volume 230 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:13, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops-dev.dagstuhl.de/entities/document/10.4230/LIPIcs.ITC.2022.13, doi:10.4230/LIPIcs.ITC.2022.13.

[RS62] J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6(1):64 – 94, 1962. doi:10.1215/ijm/1255631807.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18.

[Sui24] Sui. Sui documentation. https://docs.sui.io/guides/developer/cryptography/groth16, 2024.

[TZ23] Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 691–721. Springer, Cham, April 2023. doi:10.1007/978-3-031-30589-4_24.

[WJ13] Samuel S. Wagstaff Jr. *The Joy of Factoring*. American Mathematical Society,, Providece, Rhode Island, USA, 2013.

[Zha22] Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Cham, August 2022. doi:10.1007/978-3-031-15982-4_3.

[ZSS04] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 277–290. Springer, Berlin, Heidelberg, March 2004. doi:10.1007/978-3-540-24632-9_20.

A Related Works

The groundwork for the dBM reduction was laid out by den Boer [den90]. He studied DL and CDH in the group $\mathbb{F}_p^*$, where the Euler's totient function[8] $\phi(p-1)$ is smooth and shows that DL $\Rightarrow$ CDH in this group. It is relatively easy to generalize this approach to arbitrary groups $\mathbb{G}$ of order p, where $p-1$ is smooth as was shown later by Maurer [Mau94]. This is also the setting we use in this work.

[8] Let $n = \prod_{i=1}^{d} p_i^{e_i} > 1$ be an integer, where p_i are distinct primes. The Euler's totient function is defined as $\phi(n) = \prod_{i=1}^{d} p_i^{e_i-1}(p_i - 1)$.

Maurer [Mau94] takes the main idea of den Boer even further than that and shows how to apply it to arbitrary cyclic groups $\mathbb{G}$, where the order $p = |\mathbb{G}|$ is known. In fact, he even considers groups with non-prime orders. He proposes mapping the discrete logarithm challenge to a suitable auxiliary group with a smooth order, where the discrete logarithm can be solved in polynomial time. In particular, he focuses on using elliptic curves over $\mathbb{F}_p$ as an auxiliary group and conjectures that sufficiently smooth elliptic curve groups exist for every prime p. There are no known efficient algorithms for finding such curves for arbitrary p and many subsequent works focus on finding these elliptic curves for specific groups. The original work by den Boer can be seen as the case where $\mathbb{F}_p^*$ is the auxiliary group.

Maurer and Wolf [MW96] present various extension results. They consider different types of oracles, such as the squaring oracle and a CDH oracle with imperfect correctness. They also present a list of expressions, such as $p-1$ or $p+1$ being smooth (in fact, several more), which are sufficient for obtaining a fast reduction. They also propose how to construct groups, where DL and CDH are equivalent. The later publication by Maurer and Wolf [MW99] seems mostly to be a merger of [Mau94] and [MW96] with small extensions and more detailed proofs.

Muzereau, Smart, and Vercauteren [MSV04] present a concretely efficient version of dBM reduction and search for smooth elliptic curves for many standard elliptic curves of that time period. They have varying degrees of success, with curves for some groups having less than 40 bits smoothens and some having above 80 bits. Bentahar [Ben05] proposes an approach to optimize elliptic curve operations in dBM reduction and finds smooth elliptic curves for even more standard groups. The recent work by May and Schneider [MS23] continues this line of work. They focus on finding smooth auxiliary elliptic curve groups for many currently popular groups. Remarkably, they implement dBM reduction and achieve impressive concrete efficiency (assuming the oracle runs in constant time). For instance, for the popular NIST P-256 group, it takes around 30 seconds to break DL on modest hardware. In fact, the memory consumption seems to be the actual bottleneck in their work, e.g., 3TB in the case of P-256.

B Additional Preliminaries

B.1 SCDH = CDH

Lemma 3 ([MW96])**.** *The* CDH *assumption is hard if and only if the* SCDH *assumption is hard.*

Sketch. $\Rightarrow$) Suppose $[\![x]\!]$, $[\![y]\!]$ is the CDH challenge. If SCDH is easy to solve, the reduction can compute $[\![(x+y)^2]\!] = [\![x^2 + 2xy + y^2]\!]$, $[\![x^2]\!]$ and $[\![y^2]\!]$. Therefore,

$$(1/2) \cdot \Big([\![(x+y)^2]\!] - [\![x^2]\!] - [\![y^2]\!] \Big) = [\![xy]\!] \ ,$$

which breaks the CDH assumption.

$\Leftarrow$) Suppose $[\![x]\!]$ is the SCDH challenge and CDH is easy to solve. Then, we can sample $r_1, r_2 \leftarrow\!\!\$\ \mathbb{F}_p$ and compute uniformly random and independent group elements $[\![a]\!] := [\![x + r_1]\!]$ and $[\![b]\!] := [\![x + r_2]\!]$. Now we can use the CDH oracle to compute $[\![ab]\!] = [\![x^2 + (r_1 + r_2)x + r_1 r_2]\!]$. Therefore, $[\![ab]\!] - (r_1 + r_2)[\![x]\!] - [\![r_1 r_2]\!] = [\![x^2]\!]$, which breaks the SCDH assumption. □

B.2 Algebraic Group Model

In the algebraic group model (AGM) [FKL18], we consider security against algebraic algorithms $\mathcal{A}$. An algebraic algorithm $\mathcal{A}$ takes some group elements $[\![x_1, \ldots, x_{\mathsf{n}}]\!]$ and

possibly other bitstrings as an input. For each group element $[\![y]\!]$ that $\mathcal{A}$ outputs, it also has to output integers $a_1, \ldots, a_{\mathsf{n}}$, which satisfy $[\![y]\!] = \sum_{i=1}^{\mathsf{n}} a_i [\![x_i]\!]$. The proofs in the algebraic group model are typically reductions to some security assumption, such as the n-PDL assumption. We refer to [FKL18] for some examples. Recently, Jaeger and Mohan [JM24] gave a more rigorous formalization of AGM. Since AGM is not the main focus of this paper, we follow the intuitive formalism explained above [FKL18].

C Proof of Lemma 1

Proof. We know that $\mathbb{F}_p^*$ contains $\phi(p-1) = \prod_{i=1}^{d} p_i^{e_i - 1}(p_i - 1)$ generators, where ϕ is the Euler's totient function. According to Rosser and Schoenfeld [RS62, Eq.(3.42)], for any $n \geq 3$,

$$n/\phi(n) < e^c \ln\ln n + 2.5063/(\ln\ln n) \ ,$$

where c is the Euler's constant, e is the Euler's number, and $e^\gamma < 1.7811$. It is simple to experimentally verify that for $n \geq 2^{43} - 1$, $e^\gamma \ln\ln n + 2.5063/(\ln\ln n) < 2\ln\ln n$. Let η denote the probability that $g \leftarrow\!\!\$\ \mathbb{F}_p^*$ is a generator of $\mathbb{F}_p^*$. Then,

$$\eta = \phi(p-1)/(p-1) > 1/(2\ln\ln(p-1)) \ .$$

Recall that $\sum_{i=0}^{\infty} r^i = \frac{1}{1-r}$ for any $|r| < 1$. Therefore,

$$\sum_{i=1}^{\infty} i r^{i-1} = \frac{d}{dr} \sum_{i=0}^{\infty} r^i = \frac{d}{dr}\left(\frac{1}{1-r}\right) = \frac{1}{(1-r)^2} \ .$$

And, by taking $r = 1 - \eta < 1$,

$$\mathbb{E}[X] = \sum_{i=1}^{\infty} i(1-\eta)^{i-1}\eta = 1/\eta \ .$$

We obtain that $\mathbb{E}[X] < 2\ln\ln(p-1)$. □

D Proof of Theorem 4

Proof. Let R be a τ-time reduction algorithm in $\mathsf{Game}^{\text{gdhe-ggm}}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda)$ and

$$\mathcal{S}_m := [1 : m-1] \cup [m+1 : 2m] \ .$$

We follow a similar proof strategy as in Theorem 2. In Fig. 5, the hybrid game $\mathsf{Game}^{\text{gdhe-hyb}}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda)$ is identical to $\mathsf{Game}^{\text{gdhe-ggm}}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda)$ except for the behaviour of oracles $\overline{\mathsf{Add}}$, $\overline{\mathsf{Enc}}$, and GDHE. Namely, the oracles operate on polynomials in variable X instead of the challenge value x. Therefore, the inputs to R do not depend on x, which means that

$$\Pr[\mathsf{Game}^{\text{gdhe-hyb}}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda)] \leq 1/p \ .$$

Immediately after R receives the initial input, $\vec{f} = (1, X, \ldots, X^{\mathsf{n}})$ (observe that $\vec{f}$ is updated inside the $\overline{\mathsf{Enc}}$ oracle). As before, let us denote $\deg(\vec{f}) = \max_{g \in \vec{f}} \deg(g)$. Addition queries do not increase the degree of polynomials in $\vec{f}$, that is $\deg(\vec{f})$ remains the same before and after any $\overline{\mathsf{Add}}$ query. Suppose $\xi_0, \ldots, \xi_{m-1}, \xi_{m+1}, \ldots, \xi_{2m}$ is the first GDHE query and inputs correspond to polynomials

$$g(X), h_1(X), \ldots, h_{m-1}(X), h_{m+1}(X), \ldots, h_{2m}(X) \ .$$

The oracle defines $w(X) = h_1(X)/g(X)$ and tests that $f_i(X) = w^j(X)g(X)$ for $j \in \mathcal{S}_m$ and $i := \mathcal{I}(\xi_j)$. Recall, as in the previous impossibility proof, $\mathcal{I}(\xi_j)$ returns the smallest

index i such that $\sigma_i = \xi_j$ and $f_i(X)$ is the corresponding polynomial. If the tests pass, GDHE oracle outputs $\mathsf{Enc}(f_m(X))$, where $f_m(X) := w^m(X)g(X)$.

First, we confirm that $f_m(X)$ is a polynomial. Observe that

$$f_{m-1}(X) \cdot f_{m+1}(X) = w^{2m}(X)g^2(X) = f_m^2(X) \ ,$$

which is a polynomial since $f_{m-1}(X)$ and $f_{m+1}(X)$ are polynomials. Since $f_m^2(X) = w^{2m}(X)g^2(X)$ is a polynomial, so is $f_m(X) = w^m(X)g(X)$. Furthermore, since $\deg(f_m^2(X)) = \deg(f_{m-1}(X)) + \deg(f_{m+1}(X)) \le 2\mathsf{n}$, it follows that $\deg(f_m(X)) \le \mathsf{n}$. Therefore, after the first query to GDHE, $\vec{f}$ still contains polynomials of degree at most n. Note that if $\mathsf{n} < 2m$, $w(X)$ must be a constant polynomial.

We conclude that $\deg(\vec{f}) = \mathsf{n}$ when R finishes its execution since neither of the oracle calls can increase the degree.

For the final state of $\vec{f}$, we define the following event

$$\mathsf{bad} := \left(\begin{array}{c} \Big(\exists g \in \vec{f} : g(X) \neq 0 \wedge g(x) = 0\Big) \vee \\ \Big(\exists g, a, b \in \vec{f} \exists j \in \mathcal{S}_m : \begin{array}{c} g^{j-1}(X) \cdot b(X) \neq a^j(X) \wedge \\ g^{j-1}(x) \cdot b(x) = a^j(x) \end{array} \Big) \end{array} \right) \ ,$$

where x is the challenge chosen independently of $\vec{f}$. The first condition guarantees that when bad does not happen, then $x_{\mathcal{I}(\xi_0)} = 0$ if an only if $f_{\mathcal{I}(\xi_0)}(X) = 0$ on Step 12 in Fig. 5. For the second condition observe that if we define $w(X) = a(X)/g(X)$, then $g^{j-1}(X) \cdot b(X) = a^j(X)$ is equivalent to $b(X) = w(X)^j \cdot g(X)$. Therefore, if bad does not happen $x_i \neq w^j \cdot g$ if and only if $f_i(X) \neq w^j(X) \cdot g(X)$ on Step 7 in Fig. 5. The formal analysis that

$$\Pr[\mathsf{Game}^{\text{gdhe-ggm}}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda) \wedge \neg\mathsf{bad}] = \Pr[\mathsf{Game}^{\text{gdhe-hyb}}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda) \wedge \neg\mathsf{bad}] \le 1/p$$

is similar to the one in Theorem 2.

Let us bound the probability that the event bad happens. The vector $\vec{f}$ contains at most $\mathsf{n} + 1 + \tau$ polynomials. Probability that any non-zero $g \in \vec{f}$ has x as a root is bounded by $(\mathsf{n} + 1 + \tau) \cdot \mathsf{n}/p$. Suppose that for some $g, a, b \in \vec{f}$, $j \in \mathcal{S}_m$, we have $t(X) := g^{j-1}(X) \cdot b(X) - a^j(X) \neq 0$, but $t(x) = 0$. Since $\deg(t) \le \mathsf{n}j$, the probability of this event is at most $\mathsf{n}j/p$. There are $(\mathsf{n} + 1 + \tau)^3$ ways to choose g, a, and b. Thus,

$$\begin{aligned} \Pr[\mathsf{bad}] &\le \tfrac{(\mathsf{n}+1+\tau)\mathsf{n}}{p} + \textstyle\sum_{j\in\mathcal{S}_m} (\mathsf{n}+1+\tau)^3 \cdot \tfrac{\mathsf{n}j}{p} \\ &= \tfrac{(\mathsf{n}+1+\tau)\mathsf{n}}{p} + \tfrac{(\mathsf{n}+1+\tau)^3\mathsf{n}}{p} \cdot \textstyle\sum_{j\in\mathcal{S}_m} j \\ &\le \tfrac{(\mathsf{n}+1+\tau)\mathsf{n}}{p} + \tfrac{(\mathsf{n}+1+\tau)^3\mathsf{n}(1+2m)m}{p|} \ . \end{aligned}$$

We can now conclude that

$$\begin{aligned} \Pr\Big[\mathsf{Game}^{\text{gdhe-ggm}}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda) = 1\Big] &\le \Pr[\mathsf{bad}] + \Pr[\mathsf{Game}^{\text{gdhe-hyb}}_{\mathsf{Ggen},\mathsf{n},m,R}(1^\lambda) \wedge \neg\mathsf{bad}] \\ &= \mathcal{O}\Big(\tfrac{m^2\mathsf{n}(\mathsf{n}^3+\mathsf{n}^2\tau+\mathsf{n}\tau^2+\tau^3)}{p}\Big). \end{aligned}$$

□

E Proofs of Section 5

E.1 Perfect CDH Oracle Proof

In this section, we demonstrate Corollary 1. We first show how to define an $(1-\beta)$-CDH oracle, given an imperfect ε-CDH oracle and an upper bound of its success probability. Then, we show how to compute an upper bound on the success probability of any imperfect ε-CDH oracle. Finally, we use these results to prove Corollary 1.

$(1-\beta)$-$\mathsf{CDH}^{\varepsilon\text{-}\mathsf{CDH}(\cdot)}([\![1,x,y]\!],\beta,\varepsilon)$

1 : Find the smallest, positive integer μ such that Eq. (6) holds;
2 : **for** $i \in [1:\mu]$ **do**
3 : $\quad r_{a_i}, r_{b_i}, r_i \leftarrow\!\!\$\ \mathbb{F}_p$;
4 : $\quad [\![a_i]\!] \leftarrow r_{a_i}[\![1]\!] + r_i[\![x]\!]; [\![b_i]\!] \leftarrow r_{b_i}[\![1]\!] + r_i[\![y]\!]$;
5 : $\quad [\![u_i']\!] \leftarrow \varepsilon\text{-}\mathsf{CDH}([\![1, a_i, b_i]\!])$;
6 : $\quad [\![u_i]\!] \leftarrow \dfrac{[\![u_i']\!] - (r_{a_i} r_{b_i})[\![1]\!] - r_{a_i} r_i [\![y]\!] - r_{b_i} r_i [\![x]\!]}{r_i^2}$;
7 : **endfor**
8 : **for** $i \in [1:\mu-1], j \in [i+1:\mu]$ **do**
9 : $\quad$ **if** $[\![u_i]\!] = [\![u_j]\!]$ **then**
10 : $\quad\quad$ **return** $[\![u_j]\!]$; **endif**
11 : **endfor**
12 : **return** $\perp$;

Figure 6: Construction of a $(1-\beta)$-CDH oracle, given a ε-CDH oracle.

Theorem 6. *Let ε-CDH be any oracle such that $\mathsf{Adv}^{\mathsf{cdh}}_{\varepsilon\text{-}\mathsf{CDH}}(\lambda) \geq \varepsilon$. For each $\beta \in (0,1)$, there exists an algorithm $(1-\beta)$-CDH such that*

$$\Pr\left[\ u \neq xy \ \middle|\ \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Ggen}(1^\lambda); x, y \leftarrow\!\!\$\ \mathbb{F}_p; \\ [\![u]\!] \leftarrow (1-\beta)\text{-}\mathsf{CDH}^{\varepsilon\text{-}\mathsf{CDH}(\cdot)}(\mathsf{p}, [\![1,x,y]\!], \beta, \varepsilon); \end{array} \right] \leq \beta \ .$$

Moreover, $(1-\beta)$-CDH makes $\mathcal{O}(1/\varepsilon)\log(1/\beta)$ queries to the ε-CDH oracle, and runs in time $\mathcal{O}(1/\varepsilon)\log(1/\beta)$.

Proof. We start describing the algorithm $(1-\beta)$-CDH, which is depicted in Fig. 6. In a nutshell, the idea in [Sho97] consists of calling μ times the imperfect ε-CDH oracle, and storing the answers in a list $\vec{u} = \{[\![u_1]\!], \ldots, [\![u_\mu]\!]\}$. μ is chosen such that with high probability, the list $\vec{u}$ will contain the correct CDH answer at least twice. However, the ε-CDH oracle is called using inputs that are masked by fresh randomizers. Consequently, two oracle answers will pass the test on Step 9 only if they are the correct answers to the CDH problem, except with negligible probability. Thus, $(1-\beta)$-CDH checks whether two answers satisfy the condition on Step 9. If this happens, then, except with negligible probability, both of them are the correct CDH answers.

More formally, note that the event of getting a wrong CDH answer ($u \neq xy$) can happen only in two cases. The first case is the event that for some indexes i, j, the condition in Step 9 is satisfied ($[\![u_i]\!] = [\![u_j]\!]$), but $[\![u_j]\!] \neq [\![xy]\!]$. We call this event inc because it corresponds to the undesirable event that $(1-\beta)$-CDH returns an answer that is believed to be the correct, but it is not. The second case happens if the list $\vec{u}$ contains less than two corrects CDH evaluations, and inc did not occur. Let null be the event that there exists at most one index $i \in [1:\mu]$ such that $u_i = xy$. In this case, given inc did not occur, the algorithm reaches Step 12 and returns $\perp$. By the union bound, the probability of returning a wrong CDH answer is at most $\Pr[\mathsf{inc}] + \Pr[\mathsf{null}]$.

Intuitively, by increasing μ, $\Pr[\mathsf{null}]$ decreases, while $\Pr[\mathsf{inc}]$ increases. However, since the field dimension is exponential in the security parameter, the increase in the latter is much slower than the decrease in the former. We give an expression for $\Pr[\mathsf{null}] + \Pr[\mathsf{inc}]$ as a function of μ and p, and we impose it to be at most β.

We show in Lemma 4 that each check can lead to inc with probability at most $2/p$. Since the algorithm performs at most $\mu(\mu-1)/2$ checks, by the union bound, we have $\Pr[\mathsf{inc}] \leq \mu(\mu-1)/p$.

To bound $\Pr[\mathsf{null}]$, we note that the number of elements $[\![u_i]\!]$ such that $u_i \neq xy$ has Binomial distribution with parameters μ and ψ, where $\psi \leq 1-\varepsilon$. Let X denote the number of incorrect answers returned by ε-CDH. We have that null happens if all the μ answers are incorrect, except at most one,

$$\begin{aligned}\Pr[\mathsf{null}] &= \Pr[X \geq \mu - 1] = (\psi)^{\mu-1}(\psi + \mu(1-\psi)) \\ &\leq (1-\varepsilon)^{\mu-1}(1+\mu) \ .\end{aligned}$$

Considering the bound that we have found, we get

$$\Pr[\mathsf{null}] + \Pr[\mathsf{inc}] \leq (1-\varepsilon)^{\mu-1}(1+\mu) + \tfrac{\mu^2-\mu}{p} \ . \tag{6}$$

Therefore, the statement is satisfied as long as the chosen μ in Eq. (6) is bounded by β.

We show now that we achieve the asymptotic bounds claimed in the statement. As done in [Sho97], since p is exponential in the security parameter, we assume that the second addendum on the right side of the inequality Eq. (6) is asymptotically negligible. We want to find a range for μ such that

$$(1-\epsilon)^{\mu-1}(\mu+1) \leq \beta - \mathsf{negl}(\lambda) := \beta' \ .$$

By multiplying both sides by $(1-\epsilon)^2 \geq 0$ we get

$$(1-\epsilon)^{\mu+1}(\mu+1) \leq \beta'(1-\epsilon)^2,$$

and next, we multiply by $\ln(1-\epsilon) < 0$, which gives

$$\ln(1-\epsilon)(\mu+1)e^{\ln(1-\epsilon)(\mu+1)} \geq \beta' \ln(1-\epsilon)(1-\epsilon)^2. \tag{7}$$

The solutions for the equality in Eq. (7) are given by the Lambert W function. It is known that the equation $ye^y = x$ can be solved if and only if $x \geq -1/e$ and the solution is $y = W_0(x)$, if $x \geq 0$, or $y = W_0(x)$ and $y = W_{-1}(x)$ if $-1/e \leq x < 0$. In the case of Eq. (7), we identify $y = \ln(1-\epsilon)(\mu+1)$ and $x = \beta' \ln(1-\epsilon)(1-\epsilon)^2$. Observe that we have a solution if and only if

$$\beta' \ln(1-\epsilon)(1-\epsilon)^2 \geq -\tfrac{1}{e} \Rightarrow \beta' \leq -\tfrac{1}{e \ln(1-\epsilon)(1-\epsilon)^2} \ .$$

Here the implication follows from $\ln(1-\epsilon) < 0$. This restriction is reasonable in the context of the asymptotic behavior, where we care about small values of β'.

Hence, assuming $-1/e \leq \beta' \ln(1-\epsilon)(1-\epsilon)^2 < 0$, the solutions for the equality in Eq. (7) for $\mu \in \mathbb{R}$ are given by

$$\begin{aligned}\ln(1-\epsilon)(\mu+1) &= W_0(\beta' \ln(1-\epsilon)(1-\epsilon)^2), \quad \text{or} \\ \ln(1-\epsilon)(\mu+1) &= W_{-1}(\beta' \ln(1-\epsilon)(1-\epsilon)^2),\end{aligned}$$

given that $W_0(x) = g^{-1}(x)$, where $g : [-1, +\infty) \to [-1/e, +\infty)$, $g(x) = xe^x$ and $W_{-1}(x) = h^{-1}(x)$, where $h : (-\infty, -1] \to [-1/e, 0)$, $h(x) = xe^x$.

Thus, since we are interested in the inequality, we must have

$$\mu \leq \tfrac{W_0(\beta' \ln(1-\epsilon)(1-\epsilon)^2)}{\ln(1-\epsilon)} - 1 \text{ or } \mu \geq \tfrac{W_{-1}(\beta' \ln(1-\epsilon)(1-\epsilon)^2)}{\ln(1-\epsilon)} - 1.$$

Finally, given our focus on the positive values of μ, we consider the lower bound $\mu \geq W_{-1}(\beta' \ln(1-\epsilon)(1-\epsilon)^2)/\ln(1-\epsilon) - 1$, see Fig. 7.

Since $W_{-1}(x) \geq \ln(-x)e/(e-1)$, [Laj22, Eq. (8)], we have

$$\begin{aligned}\mu &\geq \tfrac{W_{-1}(\beta' \ln(1-\epsilon)(1-\epsilon)^2)}{\ln(1-\epsilon)} - 1 \\ &\geq \left(\tfrac{e}{e-1}\right) \tfrac{\ln\left(-\beta' \ln(1-\varepsilon)(1-\varepsilon)^2\right)}{\ln(1-\epsilon)} - 1.\end{aligned}$$

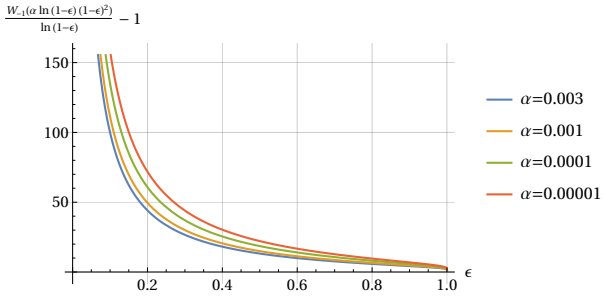

Figure 7: The Lambert W function $\frac{W_{-1}(\beta \ln(1-\epsilon)(1-\epsilon)^2)}{\ln(1-\epsilon)} - 1$ for distinct values of β.

Note that $f(X) := \ln(1-X)(1-X)^2$ is a continuous function in $(0,1)$. By analyzing the derivatives of $f(X)$, i.e., $f'(X) = (X-1) - 2(1-X)\ln(1-X)$ and $f''(X) = 3 + 2\ln(1-X)$, we find that it has a minimum at $x = 1 - 1/\sqrt{e}$. Therefore, we define $-c = f(1-1/\sqrt{e}) \approx -0.184$ and $-c \leq \ln(1-\varepsilon)(1-\varepsilon)^2$ for all $\varepsilon \in (0,1)$. Consequently

$$\mu \geq \left(\frac{e}{e-1}\right) \frac{\ln(\beta' c)}{\ln(1-\epsilon)} - 1 \ .$$

From the standard inequality $\ln(x) \leq x - 1$ for $x > 0$ and the fact that $1 - \epsilon > 0$, we have that $\ln(1-\varepsilon) \leq -\varepsilon$. Thus,

$$\begin{aligned} \mu &\geq \left(\frac{e}{e-1}\right) \frac{-\ln(\beta') - \ln(c)}{\varepsilon} - 1 \\ &= \left(\frac{e}{e-1}\right) \frac{\ln(1/\beta') + \ln(1/c)}{\varepsilon} - 1 \in \mathcal{O}(1/\varepsilon)\log(1/\beta) \ , \end{aligned}$$

where here we used the fact that $\beta \approx \beta'$.

The number of operations is clearly bounded by the number of times the check on Step 9 is performed. If we perform the checks in the naive way as depicted in Fig. 6, then we have $\mathcal{O}(1/\varepsilon^2)\log(1/\beta)$ operations, in the worst case. However, we can optimize this step using a hash table. Particularly, while we iterate on $\vec{u}$, we put the group elements $[\![u_j]\!]$ into a hash table. Then, if a collision occurred (i.e., in case we are trying to put an element in a cell of the hash table that is already occupied), we have found a pair that satisfies the equality on Step 9, and we can return it. This procedure is linear, resulting in running time for CDH of order $\mathcal{O}(1/\varepsilon)\log(1/\beta)$, as claimed in the statement. □

Even if we recover the same asymptotic behavior, differently from [Sho97], we showed here how to choose the best μ, given the arbitrary parameter β, and the field size p, to minimize the calls to the original ε-CDH oracle. One can find such optimal μ, by applying numerical methods to solve Eq. (6).

Lemma 4. *The probability that the condition on line 9 in Fig. 6 is satisfied but $[\![u]\!] \neq [\![xy]\!]$, is bounded by $2/p$.*

Proof. Using the notation Fig. 6, observe that $a_i b_i = (r_{a_i} + r_i x)(r_{b_i} + r_i y)$, which implies that $-(r_{a_i} r_{b_i}) - r_{a_i} r_i y - r_{b_i} r_i x = u'_i - a_i b_i + r_i^2 xy$. Assume that the condition on line 9 in Fig. 6 is satisfied. Thus, there exist two different indexes i, j such that $[\![u_i]\!] = [\![u_j]\!]$, and consequently

$$\begin{aligned} r_i^2 u_j = r_i^2 u_i \quad &= u'_i - (r_{a_i} r_{b_i}) - r_{a_i} r_i y - r_{b_i} r_i x \\ &= u'_i - a_i b_i + r_i^2 xy \ . \end{aligned}$$

This can be rewritten as

$$r_i^2 (u_j - xy) + (r_{a_i} r_{b_i} - u'_i) = 0 \ .$$

$\mathsf{Est}^{\varepsilon\text{-}\mathsf{CDH}}(\beta, c)$

1 : $L \leftarrow 0; \mu \leftarrow 1 + \lceil \ln(2/\beta)/2c^2 \rceil$;
2 : **for** $i \in [1 : \mu]$ **do**
3 : $\quad x, y \leftarrow\$ \mathbb{F}_p$;
4 : $\quad [\![u]\!] \leftarrow \varepsilon\text{-}\mathsf{CDH}([\![1, x, y]\!])$;
5 : $\quad$ **if** $[\![u]\!] = xy[\![1]\!]$ **then** $L \leftarrow L + 1$;
6 : **endfor**
7 : $\varepsilon' \leftarrow L/\mu$; **return** ε';

Figure 8: Est estimates the probability of success of any faulty ε-CDH oracle.

Since the inputs to ε-CDH were randomized, all of u_j, x, y, r_{a_i}, r_{b_i}, and u'_i do not depend on r_i. Consider the polynomial $w(X) := (u_j - xy)X^2 + r_{a_i} r_{b_i} - u'_i$. If $u_j \neq xy$, then $w(X) \neq 0$. Thus, the probability that $w(r_i) = 0$ for a random $r_i \leftarrow\$ \mathbb{F}_p$ is at most $2/p$. □

We show next how to bound $\mathsf{Adv}^{\mathsf{cdh}}_{\mathcal{A}}(\lambda)$, for each CDH adversaries.

Lemma 5. *Let ε-CDH be an oracle such that $\mathsf{Adv}^{\mathsf{cdh}}_{\varepsilon\text{-}\mathsf{CDH}}(\lambda) \geq \varepsilon$. For any β, c such that $0 < \beta, c < 1$, we have that*

$$\Pr\left[\ |\varepsilon - \varepsilon'| \geq c \ \middle|\ \varepsilon' \leftarrow \mathsf{Est}^{\varepsilon\text{-}\mathsf{CDH}}(\beta, c);\ \right] \leq \beta \ ,$$

where $\mathsf{Est}^{\varepsilon\text{-}\mathsf{CDH}}(\beta, c)$ is the algorithm defined in Fig. 8.

Proof. Let us start by recalling Hoeffding's inequality when applied to a sum of independent Bernoulli's random variables. Particularly, let $\{X_i\}$ be a sequence of independent Bernoulli's trials and let $L = \sum_{i=1}^{\mu} X_i$. For each $t > 0$, we have

$$\Pr\left(|L - \mathbb{E}(L)| \geq t\right) \leq 2\exp(-2t^2/\mu) \ .$$

We will consider $X_i = 1$ if the check on the line 5, at the i-th iteration, is satisfied.

Note that $\mathbb{E}(X_i) \leq \varepsilon$, for each $i \in [1 : \mu]$. Conditioned on the event that ε' is returned by $\mathsf{Est}^{\varepsilon\text{-}\mathsf{CDH}}$, on input any given β, c, we have that

$$\Pr(|\varepsilon' - \varepsilon| \geq c) \leq \Pr(|L - \varepsilon\mu| \geq c\mu) \leq 2\exp(-2c^2\mu) \ .$$

Here, the equality follows from the definition of ε' and the linearity of the expected value, while the inequality follows from Hoeffding's inequality.

Now, we want that $2\exp(-2c^2\mu) \leq \beta$, which is satisfied if $\mu \geq \ln(2/\beta)/(2c^2)$. □

Finally, applying both Lemma 5 and Theorem 6 we can prove Corollary 1. For the sake of the reader, we restate it here.

Corollary 3. *Let $\beta \in (0, 1)$ be an arbitrary constant. Suppose the CDH assumption does not hold. Thus, there exists a PPT $\mathcal{B}$ and a polynomial $s(X)$, such that, there exists an integer λ_0, such that $\mathsf{Adv}^{\mathsf{cdh}}_{\mathcal{B}}(\lambda) \geq 1/s(\lambda)$, for each $\lambda \geq \lambda_0$. Then, there exists a CDH adversary $\mathcal{A}$ with the following properties.*

1. $\Pr\left[\ u \neq x, y \ \middle|\ x, y \leftarrow\$ \mathbb{F}_p; [\![u]\!] \leftarrow \mathcal{A}([\![1, x, y]\!], \beta);\ \right] \geq 1 - \beta - 2^{-\lambda}$.
2. *$\mathcal{A}$ is expected PPT.*
3. *$\mathcal{A}$ makes $1 + 50(\lambda - 1) + \mathcal{O}(1/\varepsilon)\log(1/\beta)$ calls, on average, to the adversary $\mathcal{B}$. Only the last $\mathcal{O}(1/\varepsilon)\log(1/\beta)$ calls depend on the input $[\![x, y]\!]$.*

$\mathcal{A}^{\mathcal{B}}([\![1, x, y]\!], \beta, \beta')$

1 : $c \leftarrow 1/10; \varepsilon' \leftarrow \mathsf{Est}^{\mathcal{B}}(2^\lambda, 1/10);$

2 : $[\![u]\!] \leftarrow (1-\beta)\text{-}\mathsf{CDH}^{\mathcal{B}}([\![1, x, y]\!], \beta, \varepsilon' + c);$

3 : **return** $[\![u]\!];$

Figure 9: An adversary $\mathcal{A}$ that solves CDH with arbitrary high probability.

Proof. We define $\mathcal{A}$ having black-box access to $\mathcal{B}$ in Fig. 9. $\mathcal{A}$ first calls the procedure Est to estimate $\mathsf{Adv}^{\mathsf{cdh}}_{\mathcal{B}}(\lambda) \geq \varepsilon$. Then it runs the amplified oracle $(1-\beta)$-CDH, defined in Fig. 6, and forwards its output. Clearly, if both Est and $(1-\beta)$-CDH are successful, then $\mathcal{A}$ returns the correct answer. Thus, by the union bound, Lemma 5, and Theorem 6 we have $\mathsf{Adv}^{\mathsf{cdh}}_{\mathcal{A}}(\lambda) \geq 1 - \beta - 1/2^\lambda$.

To check the statement about the running time, note that Est runs in time $1 + \left\lceil \frac{\log(2/\beta)}{2(1/10)^2} \right\rceil$. The choice of $c = 1/10$ is arbitrary and the proof does not depend on this choice. Let null be the event that $\varepsilon' > 1/s(\lambda) + 1/10$. Clearly, null can happen only if Est fails. In case Est is successful (i.e., null did not occur), then $\varepsilon' + c$ is polynomial in the security parameter, and, by Theorem 6, $(1-\beta)$-CDH running time is $\mathcal{O}(1/(\varepsilon'+c)) \log(1/\beta) = \mathcal{O}(1/\varepsilon) \log(1/\beta)$.

To ensure that $\mathcal{A}$ is the expected PPT, we will modify $(1-\beta)$-CDH such that it aborts if it has to perform more than 2^λ operation. Thus when null happens, $\mathcal{A}$ runs in time $\mathcal{O}(2^\lambda)$. Therefore, the expected running time of $\mathcal{A}$ is bound by $\mathcal{O}(1/\varepsilon) \log(1/\beta) + \mathcal{O}(1)$.

The third claim follows trivially from Lemma 5, and Theorem 6, once we note that only the former needs the inputs $[\![x, y]\!]$. □

Note that if we want to use $\mathcal{A}$ several times, we do not need to estimate $\mathsf{Adv}^{\mathsf{cdh}}_{\mathcal{B}}(\lambda)$ more than once. In fact, since this step is independent of the specific CDH challenge, we can perform it once and then just call the second procedure.

E.2 Perfect DHE Oracle Proof

In this section, we sketch a proof of Corollary 2. The procedure to obtain an upper bound for the advantage of ε-DHE is similar to the one defined in Lemma 5 for CDH. In particular, the reduction challenges the adversary with fresh, random instances. Since the reduction knows the trapdoors of these challenges, it can verify whether the adversary succeeded, and it sets the success probability as the fraction of successes. Thus, we only care about defining a $(1-\beta)$-CDH oracle, given an imperfect ε-DHE oracle and an upper bound of its success probability. We start with a lemma for the DHE problem that plays a similar role to Lemma 4 for the CDH.

Lemma 6. *Let $f(X, A, B)$ be the polynomial such that $(AX + B)^{\mathsf{n}+1} = A^{\mathsf{n}+1}X^{\mathsf{n}+1} + f(X, A, B)$, and Let $\mathcal{A}$ be any* n-DHE *adversary. We have that*

$$\Pr\left[\begin{array}{l|l} u_1 = u_2; & a_1, a_2, b_1, b_2, x \leftarrow\!\!\$\ \mathbb{F}_p; \\ u_1 \neq x^{\mathsf{n}+1}; & [\![u_1']\!] \leftarrow \mathcal{A}([\![\{(a_1 x + b_1)^i\}_{i=0}^{\mathsf{n}}]\!]); \\ & [\![u_2']\!] \leftarrow \mathcal{A}([\![\{(a_2 x + b_2)^i\}_{i=0}^{\mathsf{n}}]\!]); \\ & [\![u_1]\!] \leftarrow ([\![u_1']\!] - [\![f(x, a_1, b_1)]\!])/a_1^{\mathsf{n}+1}; \\ & [\![u_2]\!] \leftarrow ([\![u_2']\!] - [\![f(x, a_2, b_2)]\!])/a_2^{\mathsf{n}+1}; \end{array}\right] \leq \frac{2(\mathsf{n}+1)}{p} \ .$$

Proof. Consider the polynomial $g(A) = A^{\mathsf{n}+1}u_1 - u_2' + f(x, A, b_2)$. From $u_1 = u_2$, we get

$$u_1 = u_2 = \frac{u_2' - f(x, a_2, b_2)}{a_2^{\mathsf{n}+1}} \ .$$

$(1-\beta)$-$\mathsf{DHE}^{\varepsilon\text{-}\mathsf{DHE}(\cdot)}([\![\{x^i\}_{i=0}^{\mathsf{n}}]\!], \beta, \varepsilon)$

1 : Find the smallest, positive integer μ such that Eq. (8) holds;
2 : **for** $i \in [1:\mu]$ **do**
3 : $\quad r_{a_i}, r_{b_i} \leftarrow\!\$\ \mathbb{F}_p$;
4 : $\quad$ **for** $j \in [0:\mathsf{n}]$ **do** $[\![y_i^j]\!] \leftarrow [\![(a_i x + b_i)^j]\!]$; **endfor**
5 : $\quad [\![u_i']\!] \leftarrow \varepsilon$-$\mathsf{DHE}([\![\{y^i\}_{i=0}^{\mathsf{n}}]\!])$;
6 : $\quad [\![u_i]\!] \leftarrow \dfrac{[\![u_i']\!] - [\![f(x, a_i, b_i)]\!]}{a_i^{\mathsf{n}+1}}$;
7 : **endfor**
8 : **for** $i \in [1:\mu-1], j \in [i+1:\mu]$ **do**
9 : $\quad$ **if** $[\![u_i]\!] = [\![u_j]\!]$ **then**
10 : $\quad\quad$ **return** $[\![u_j]\!]$; **endif**
11 : **endfor**
12 : **return** $\perp$;

Figure 10: Construction of a $(1-\beta)$-DHE oracle, given a ε-DHE oracle.

Thus, $u_1 = u_2$ implies $g(a_2) = 0$.

Suppose that $u_1 \neq x^{\mathsf{n}+1}$ and $u_2 = x^{\mathsf{n}+1}$, which implies that $u_2' = (a_2 x + b_2)^{\mathsf{n}+1}$, $g(A) \neq 0$ as polynomial, and $g(a_2) = 0$. We can apply the Schwartz–Zippel lemma and argue that the previous event happens at most with probability $(\mathsf{n}+1)/|\mathbb{F}_p|$, since a_2 is information-theoretically hidden to $\mathcal{A}$. If $u_2 \neq x^{\mathsf{n}+1}$, then trivially $g(A) \neq 0$ as polynomial, and $g(a_2) = 0$. So again, this can happen at most with probability $(\mathsf{n}+1)/|\mathbb{F}_p|$. The claim follows from the union bound. □

Theorem 7. *Let* ε-DHE *be any oracle such that* $\mathsf{Adv}^{\mathsf{n},\mathsf{dhe}}_{\varepsilon\text{-}\mathsf{DHE}}(\lambda) \geq \varepsilon$. *For each* $\beta \in (0,1)$, *there exists an algorithm* $(1-\beta)$-DHE *such that*

$$\Pr\left[\ u \neq x^{\mathsf{n}+1} \ \middle|\ \begin{array}{l} \mathsf{p} \leftarrow \mathsf{Ggen}(1^\lambda); x \leftarrow\!\$\ \mathbb{F}_p; \\ [\![u]\!] \leftarrow (1-\beta)\text{-}\mathsf{CDH}^{\varepsilon\text{-}\mathsf{CDH}(\cdot)}(\mathsf{p}, [\![\{x^i\}_{i=1}^{\mathsf{n}}]\!], \beta, \varepsilon); \end{array} \right] \leq \beta \ .$$

Moreover, $(1-\beta)$-DHE *makes* $\mathcal{O}(1/\varepsilon)\log(1/\beta)$ *queries to the* ε-CDH *oracle, and runs in time* $\mathcal{O}(1/\varepsilon)\log(1/\beta)$.

Sketch. The proof is essentially the same as the one in Theorem 6. Therefore, we only give a sketch, outlining the different parts. Recall that the algorithm $(1-\beta)$-DHE can return the wrong answer only in two cases.

The first one is that the check on line 9 is satisfied, but the returned element is not the correct answer. We call this event inc (false positive). We invoke Lemma 6 to claim that each check can lead to inc with probability at most $2\mathsf{n}/p$. Since there are at most $(\mu^2-\mu)/2$ checks, we have that

$$\Pr[\mathsf{inc}] \leq \frac{(\mathsf{n}+1)(\mu^2-\mu)}{p} \ .$$

The $(1-\beta)$-DHE can return the wrong answer also in the case that it reaches line 9 and it returns $\perp$. This will happen if the list $\vec{u}$ contains the correct DHE answer at most once. We call this event null. Thus, if null happens, there exists at most one index $i \in [1:\mu]$ such that $u_i = x^{\mathsf{n}+1}$. Using the same argument we have used in Theorem 6, we get that

$$\Pr[\mathsf{null}] \leq (1-\varepsilon)^{\mu-1}(1+\mu) \ .$$

Thus, the statement is satisfied as long we have

$$\Pr[\mathsf{null}] + \Pr[\mathsf{inc}] \leq (1-\varepsilon)^{\mu-1}(1+\mu) + \frac{(\mathsf{n}+1)(\mu^2-\mu)}{p} \leq \beta \ . \tag{8}$$

Once again, one can use numerical methods to solve the previous inequality and find the smallest suitable integer value for μ.

Since n is polynomial in the security parameter, the asymptotic behavior is the same as in Theorem 6. Therefore, we omit the details here. □