Tight Reductions for Diffie-Hellman Variants in the Algebraic Group Model

Taiga Mizuide*, Atsushi Takayasu†, Tsuyoshi Takagi‡

December 20, 2018

Abstract

Fuchsbauer, Kiltz, and Loss (Crypto'18) gave a simple and clean definition of an *algebraic group model (AGM)* that lies in between the standard model and the generic group model (GGM). Specifically, an algebraic adversary is able to exploit group-specific structures as the standard model while the AGM successfully provides meaningful hardness results as the GGM. As an application of the AGM, they show a tight computational equivalence between the computing Diffie-Hellman (CDH) assumption and the discrete logarithm (DL) assumption. For the purpose, they used the square Diffie-Hellman assumption as a bridge, i.e., they first proved the equivalence between the DL assumption and the square Diffie-Hellman assumption, then used the known equivalence between the square Diffie-Hellman assumption and the CDH assumption. In this paper, we provide an alternative proof that directly shows the tight equivalence between the DL assumption and the CDH assumption. The crucial benefit of the direct reduction is that we can easily extend the approach to variants of the CDH assumption, e.g., the bilinear Diffie-Hellman assumption. Indeed, we show several tight computational equivalences and discuss applicabilities of our techniques.

*Department of Creative Informatics, The University of Tokyo, Tokyo, Japan.

†Department of Mathematical Informatics, The University of Tokyo, Tokyo, Japan, and National Institute of Advanced Industrial Science and Technology, Tokyo, Japan. (takayasu@mist.i.u-tokyo.ac.jp)

‡Department of Mathematical Informatics, The University of Tokyo, Tokyo, Japan.

Contents

1 Introduction **3**
- 1.1 Background . . . 3
- 1.2 Our Contributions . . . 4
- 1.3 Organization . . . 4

2 Preliminaries **5**
- 2.1 Computational Problems . . . 5
- 2.2 Algebraic Group Model . . . 7

3 Reduction from DL in Cyclic Groups **9**
- 3.1 Basic Reduction: From DL to CDH . . . 9
- 3.2 Master Theorem in Cyclic Groups . . . 10

4 Reduction from BDL in Bilinear Groups **11**
- 4.1 Algebraic Bilinear Group Model . . . 11
- 4.2 From BDL to BDH . . . 12
- 4.3 Master Theorem in Bilinear Groups . . . 13

5 DL to k-Lin Reduction **15**

6 Conclusion **17**

1 Introduction

1.1 Background

Diffie-Hellman Problem in the Generic Group Model. The discrete logarithm (DL) assumption and the computational Diffie-Hellman (CDH) assumption including its variants have been devoted to constructing numerous cryptographic protocols. Hence, estimating the computational hardness of solving the problems is a fundamental research topic in cryptography. For the purpose, the *generic group model* (GGM) [Nec94, BL96, Sho97, MW98, Mau05] over cyclic groups is a wonderful tool and has successfully provided several fantastic results in the context. Generic algorithms are not able to exploit specific structures of cyclic groups in the sense that the algorithms are given group elements only via abstract handles. Then, the algorithms are able to output only group elements which are computed by interacting with an oracle and applying group operations to given elements. Therefore, generic algorithms such as a baby-step giant-step algorithm, the Pohlig-Hellman algorithm [PH78] (in composite-order groups), and Pollard's rho algorithm [Pol78] work in any cyclic groups.

Furthermore, the most substantial benefit of the GGM is that we are able to derive information theoretic lower bounds of computational problems, where analogous analyses seem infeasible in the standard model. For example, any generic algorithms require at least $O(\sqrt{p})$ group operations to solve the DL problem in cyclic groups of a prime-order p. Analogous analyses have also been made for the CDH problem and its variants in an ad-hoc manner. Thus far, the GGM has been extended and used for studying computational problems in bilinear (and multilinear) groups [BB08, Boy08, KSW13, MRV16, EHK+17].

One main criticism of the GGM is that computational problems that are generically hard may not be hard when instantiated in concrete groups. Jager and Schwenk [JS13] proved that computing a Jacobi symbol of an integer modulo a composite n generically is equivalent to factorization; however, the computation is easy when given an actual representation of $\mathbb{Z}_n$. Similarly, the number field sieves [Gor93] in specific groups are able to solve the DL problem in subexponential time in $\log p$, i.e., faster than the generic algorithms. Hence, the GGM gives us certain confidence of computational hardness while we want to obtain analogous results in the standard model or less restricted models than the GGM.

Algebraic Group Model. In Crypto'18, Fuchsbauer, Kiltz, and Loss [FKL18] introduced an *algebraic group model* (AGM). The definition of the AGM lies in between the standard model and the GGM. Like the standard model and unlike the GGM, an algebraic algorithm is given an actual representation of cyclic groups. On the other hand, like the GGM and unlike the standard model, an algebraic algorithm is able to output only group elements by applying group operations to given elements. Although the algebraic algorithm is not required to interact with an oracle for the computation, it should output a record of a group operation which Fuchsbauer et al. called a *representation*. Let $\mathcal{G} := (\mathbb{G}, G, p)$ be a group description, where $\mathbb{G}$ is an additive cyclic group of a prime-order p and G is a generator. When an algebraic algorithm is given $\left(\mathcal{G}, \vec{X} := (X_1, \ldots, X_\ell) \in \mathbb{G}^\ell\right)$ and outputs $Z \in \mathbb{G}$, it has to also output a vector $\vec{z} := (z_0, z_1, \ldots, z_\ell) \in \mathbb{Z}_p^{\ell+1}$ as a representation of Z with respect to $\vec{X}$ such that $Z = \sum_{i=0}^{\ell} z_i X_i$, where $X_0 := G$. Similar definitions of an algebraic algorithm are already known in [BV98, PV05]; however, Fuchsbauer et al.'s definition is simpler and clearer.

The AGM is not allowed to derive computational lower bounds as the standard model. In turn, as opposed to the standard model, Fuchsbauer et al. showed that the AGM is able to make a tight reduction from the DL to the CDH. To be precise, they used the square Diffie-Hellman (DH)

problem [MW96, BDS98] as an intermediate step. They first proved a tight reduction from the DL to the square DH in the AGM. Let $(\mathcal{G}, X)$ be a DL instance such that $X := xG$. The reduction algorithm gives $(\mathcal{G}, X)$ to a square DH algorithm and receives an answer $Z = x^2G$ along with a representation vector $\vec{z}$. Fuchsbauer et al. showed that the vector $\vec{z}$ and the relation $z_0G + z_1X = Z$ are sufficient to recover the DL solution x by solving an equation modulo a prime p. Then, due to the known computational equivalence between the square DH and the CDH [MW99, BDZ03], their reduction implies a tight reduction from the DL to the CDH in the AGM. Furthermore, a valuable feature of the result is that the reduction algorithm is *generic*. Due to the fact, an existence of the tight reduction implies an information theoretic lower bounds of the CDH as $O(\sqrt{p})$ in the GGM.

Fuchsbauer et al. claimed that a benefit of the AGM is that we are able to derive information theoretic lower bounds of the CDH in the GGM via *quite simple* arguments. Indeed, Fuchsbauer et al.'s reduction in the AGM is much simpler than the analysis in the GGM. Therefore, providing generic reductions from the DL to other computational problems of the CDH family in the AGM has to be an interesting open problem.

1.2 Our Contributions

In this paper, we provide generic and tight reductions from the DL to several computational problems of the CDH family in the AGM. A starting point of our technique is a *direct* reduction from the DL to the CDH *without* using the square DH as the intermediate step. Given the DL instance $(\mathcal{G}, X)$, our reduction algorithm randomly samples $r \in \mathbb{Z}_p$ and gives $(\mathcal{G}, (X_1, X_2))$ to a CDH algorithm, where $X_1 := X = xG$ and $X_2 := X + rG = (x + r)G$. Here, $(\mathcal{G}, (X_1, X_2))$ is a properly distributed CDH instance in the sense that x and $x + r$ are independently distributed to uniform in $\mathbb{Z}_p$ from the CDH algorithm's view. Then, the reduction algorithm receives a solution of the CDH $Z = x(x + r)G$ along with a representation vector $\vec{z}$. We show that the vector $\vec{z}$ and the relation $z_0G + z_1X_1 + z_2X_2 = Z$ are sufficient to recover x by solving an equation modulo a prime p. The approach is very simple as Fuchsbauer et al.'s one and easily applicable to several CDH variants which are not studied in [FKL18]. We believe that the simple approach is a main benefit of our result. To explain our technique as simple as possible, we consider only *tight* reductions in the sense that the reduction algorithm uses an algorithm for CDH variants only once.

Furthermore, we extend the AGM to an *algebraic bilinear group model* (ABGM) for studying computational problems in symmetric bilinear groups equipped with a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We define an algebraic bilinear algorithm so that it is given $\big(\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p), \vec{X} := (X_1, \ldots, X_k) \in \mathbb{G}^k, \vec{Y} := (Y_1, \ldots, Y_\ell) \in \mathbb{G}_T^\ell\big)$ and outputs $Z \in \mathbb{G}_T$ along with a representation vector $\vec{z}$ that indicates how Z is computed by the given elements. Then, we extend the approach used in cyclic groups and provide generic and tight reductions from the DL to several computational problems of the CDH family including the bilinear Diffie-Hellman problem.

Finally, we provide our master theorems that indicate what kind of computational assumptions can be reduced to from the DL assumption both in cyclic groups and bilinear groups of a prime-order.

We should note that the master theorem does not capture the standard k-linear assumption. Hence, we slightly modify the above approach and successfully provide a tailor-made reduction from the DL to the k-linear assumption.

1.3 Organization

In Section 2, we recall several computational problems which we study in this paper. Then, we show a definition of algebraic group models defined by Fuchsbauer et al. In Section 3 and 4, we

show our technique to provide generic and tight reductions from the DL to the CDH family in cyclic groups and bilinear groups along with a master theorem, respectively. In Section 5, we provide a tailor-made reduction from the DL to the k-linear assumption.

2 Preliminaries

In Section 2.1, we review several computational problems in cyclic groups and in bilinear groups. In Section 2.2, we recall a basic notion of security games, the generic group model, and the algebraic group model. The contents of this section heavily refer to [Boy08, KSW13, EHK+17, FKL18].

Notations. We use $x \xleftarrow{\$} \mathbb{Z}_p$ to denote a uniformly random sampling from $\mathbb{Z}_p$ and $(x_1, \ldots, x_\ell) \xleftarrow{\$} \mathbb{Z}_p^\ell$ to denote every element is sampled by $x_i \xleftarrow{\$} \mathbb{Z}_p$ independently. For a positive integer $\ell > 0$, we use $[\ell]$ to denote a set of integers $\{1, 2, \ldots, \ell\}$. For an $(m+n)$-variate polynomial $f(x_1, \ldots, x_m, y_1, \ldots, y_n)$, we use $\deg f$ to denote a degree of the polynomial and $\deg_{x_1,\ldots,x_m} f$ to denote a degree of the polynomial only with respect to variables $x_1, \ldots, x_m$. As an example for $f(x, y, z) := x^2yz$, we use the notations $\deg f = 4$, $\deg_x f = 2$, and $\deg_{x,y} = 3$.

2.1 Computational Problems

We first review computational problems in cyclic groups, then do them in bilinear groups.

Computational Problems in Cyclic Groups. We review computational problems in cyclic groups. Let $\mathcal{G} := (\mathbb{G}, G, p)$ be a group description, where $\mathbb{G}$ is an additive group generated by G and has a prime-order p.[1] We first define a discrete logarithm problem to which other problems will be reduced.

Definition 1 (Discrete Logarithm (DL) Problem)**.** *Given a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and a group element $X := xG \in \mathbb{G}; x \xleftarrow{\$} \mathbb{Z}_p$, compute $x \in \mathbb{Z}_p$.*

Then, we summarize the CDH problem and its variants which we study in this paper.

Definition 2 (Computational Diffie-Hellman (CDH) Problem [DH76])**.** *Given a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and group elements $(X_1 := x_1G, X_2 := x_2G) \in \mathbb{G}^2; (x_1, x_2) \xleftarrow{\$} \mathbb{Z}_p^2$, compute $Z := x_1x_2G \in \mathbb{G}$.*

Definition 3 (k-party Diffie-Hellman (k-PDH) Problem [Bis08])**.** *Given a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and group elements $(X_1 := x_1G, \ldots, X_k := x_kG) \in \mathbb{G}^k; (x_1, \ldots, x_k) \xleftarrow{\$} \mathbb{Z}_p^k$, compute $Z := x_1 \cdots x_kG \in \mathbb{G}$.*

The following k-exponent Diffie-Hellman assumption for $k = 2$ called the square Diffie-Hellman assumption was used in [MW96, BDS98].

Definition 4 (k-exponent Diffie-Hellman (k-EDH) Problem)**.** *Given a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and a group element $X := xG \in \mathbb{G}; x \xleftarrow{\$} \mathbb{Z}_p$, compute $Z := x^kG \in \mathbb{G}$.*

The following k-th root Diffie-Hellman problem for $k = 2$ called the square root Diffie-Hellman problem was used in [KMS04].

[1] To construct a reduction, we solve an equation modulo an order of $\mathbb{G}$. Hence, if the order is composite, we do not know how to solve it in general. Hence, as [FKL18] we study only a prime-order group in this paper.

Definition 5 (k-th Root Diffie-Hellman (k-RDH) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, G, p)$ *and a group element* $X := x^k G \in \mathbb{G}; x \xleftarrow{\$} \mathbb{Z}_p$*, compute* $Z := xG \in \mathbb{G}$*.*

Next, we recall a *decisional* k-linear problem.

Definition 6 (Decisional k-Linear Problem [BBS04])**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, G, p)$ *and group elements* $(X_1 := x_1 G, \ldots, X_k := x_k G, Y_1 := x_1 y_1 G, \ldots, Y_k := x_k y_k G, Z) \in \mathbb{G}^{2k+1}; (x_1, \ldots, x_k, y_1, \ldots, y_k) \xleftarrow{\$} \mathbb{Z}_p^{2k}$*, distinguish whether* $Z := (y_1 + \cdots + y_k)G$ *or* Z *is a random element in* $\mathbb{G}$*.*

In this paper, we define computational variants of the k-linear problem in the following two ways.

Definition 7 (Computational k-Linear (k-Lin$^{(1)}$) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, G, p)$ *and group elements* $(X_1 := x_1 G, \ldots, X_k := x_k G, Y_1 := x_1 y_1 G, \ldots, Y_k := x_k y_k G) \in \mathbb{G}^{2k+1}; (x_1, \ldots, x_k, y_1, \ldots, y_k) \xleftarrow{\$} \mathbb{Z}_p^{2k}$*, compute* $Z := (y_1 + \cdots + y_k)G$*.*

Definition 8 (Computational k-Linear (k-Lin$^{(2)}$) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, G, p)$ *and group elements* $(X_1 := x_1 G, \ldots, X_k := x_k G, Y_1 := x_1 y_1 G, \ldots, Y_{k-1} := x_{k-1} y_{k-1} G, Y' := (y_1 + \cdots + y_k)G) \in \mathbb{G}^{2k}; (x_1, \ldots, x_k, y_1, \ldots, y_k) \xleftarrow{\$} \mathbb{Z}_p^{2k}$*, compute* $Z := x_k y_k G$*.*

A natural definition of a computational variant should be k-Lin$^{(1)}$; however, our master theorem will not capture the problem. Therefore, we also define k-Lin$^{(2)}$ which is included in the master theorem. The difference may be a good example to give an intuitive understanding of our master theorem. We will later provide a tailor-made reduction for k-Lin$^{(1)}$ by slightly modifying our technique.

To provide our master theorem in cyclic groups, we define a generalized version of the Diffie-Hellman problem as follows.

Definition 9 (Generalized Diffie-Hellman (GDH) Problem)**.** *Let* $f_1(x_1, \ldots, x_m, y_1, \ldots, y_n), \ldots, f_\ell(x_1, \ldots, x_m, y_1, \ldots, y_n)$*, and* $g(x_1, \ldots, x_m)$ *be known fixed non-zero polynomials. Given a group description* $\mathcal{G} := (\mathbb{G}, G, p)$ *and group elements*

$$(X_1 := f_1(x_1, \ldots, x_m, y_1, \ldots, y_n)G, \ldots, X_\ell := f_\ell(x_1, \ldots, x_m, y_1, \ldots, y_n)G) \in \mathbb{G}^\ell;$$
$$(x_1, \ldots, x_m, y_1, \ldots, y_n) \xleftarrow{\$} \mathbb{Z}_p^{m+n},$$

compute

$$Z := g(x_1, \ldots, x_m)G.$$

Our master theorem will indicate that when the GDH problem can be reduced to from the DL.

Computational Problems in Bilinear Groups. We review computational problems in bilinear groups. For simplicity, we focus only on *symmetric* bilinear maps $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ throughout this paper. Let $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ be a bilinear group description, where $\mathbb{G}$ is an additive group generated by G and has a prime-order p, and $\mathbb{G}_T$ is a multiplicative group of order p associated with a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, i.e., $e(G, G)$ is a generator of $\mathbb{G}_T$ and $e(xG, yG) = e(G, G)^{xy}$.

We will provide a reduction from the DL in source groups $\mathbb{G}$ to CDH variants. Hence, we define a bilinear discrete logarithm problem as follows.

Definition 10 (Bilinear Discrete Logarithm (BDL) Problem)**.** *Given a bilinear group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ *and a group element* $X := xG \in \mathbb{G}; x \xleftarrow{\$} \mathbb{Z}_p$*, compute* $x \in \mathbb{Z}_p$*.*

Then, we summarize the CDH variants in bilinear groups.

Definition 11 (Bilinear Diffie-Hellman (BDH) Problem [BF03, Jou04])**.** *Given a bilinear group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ *and group elements* $(X_1 := x_1G, X_2 := x_2G, X_3 := x_3G) \in \mathbb{G}^3; (x_1, x_2, x_3) \xleftarrow{\$} \mathbb{Z}_p^3$*, compute* $Z := e(G, G)^{x_1x_2x_3} \in \mathbb{G}_T$*.*

The following ℓ-weak bilinear Diffie-Hellman inversion problem is a *parametric* problem defined with a fixed integer ℓ. Although we define the problem with general ℓ, our technique will be able to provide generic reductions from the DL problem to the parametric problem only for $\ell = 1$.

Definition 12 (ℓ-weak Bilinear Diffie-Hellman Inversion (ℓ-wBDHI) Problem [Boy08])**.** *Given a bilinear group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ *and group elements* $(X_1 := x_1G, X_2 := x_1^2G, \ldots, X_\ell := x_1^\ell G, X_{\ell+1} := x_2G) \in \mathbb{G}^{\ell+1}; (x_1, x_2) \xleftarrow{\$} \mathbb{Z}_p^2$*, compute* $Z := e(G, G)^{x_1^{\ell+1}x_2} \in \mathbb{G}_T$*.*

To provide our master theorem in bilinear groups, we define a generalized version of the bilinear Diffie-Hellman problem as follows.

Definition 13 (Generalized Bilinear Diffie-Hellman (GBDH) Problem)**.** *Let* $f_1(x_1, \ldots, x_m, y_1, \ldots, y_n), \ldots, f_k(x_1, \ldots, x_m, y_1, \ldots, y_n)$, $g_1(x_1, \ldots, x_m, y_1, \ldots, y_n), \ldots, g_\ell(x_1, \ldots, x_m, y_1, \ldots, y_n)$*, and* $h(x_1, \ldots, x_m)$ *be known fixed non-zero polynomials. Given a bilinear group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ *and group elements*

$$\left(\begin{array}{c} X_1 := f_1(x_1, \ldots, x_m, y_1, \ldots, y_n)G, \ldots, X_k := f_k(x_1, \ldots, x_m, y_1, \ldots, y_n)G, \\ Y_1 := e(G, G)^{g_1(x_1, \ldots, x_m, y_1, \ldots, y_n)}, \ldots, Y_\ell := e(G, G)^{g_\ell(x_1, \ldots, x_m, y_1, \ldots, y_n)} \end{array} \right) \in \mathbb{G}^k \times \mathbb{G}_T^\ell;$$
$$(x_1, \ldots, x_m, y_1, \ldots, y_n) \xleftarrow{\$} \mathbb{Z}_p^{m+n},$$

compute

$$Z := e(G, G)^{h(x_1, \ldots, x_m)}.$$

2.2 Algebraic Group Model

In this subsection, we review the GGM and the AGM in cyclic groups.

Algebraic Security Game. Let $\mathbf{G}_\mathcal{G}$ be an *algebraic* security game relative to a group description $\mathcal{G} := (\mathbb{G}, G, p)$; an adversary A receives $\mathcal{G}$ and an instance of the problem $\vec{X}$ from a challenger, then returns an output. For example, we use $\mathbf{CDH}_\mathcal{G}$ to denote security games of the CDH problem relative to $\mathcal{G}$; an adversary A receives $\mathcal{G}$ and (X_1, X_2) from a challenger, then returns an output Z. We use $\mathbf{G}_\mathcal{G}^\mathsf{A}$ to denote an output of a game $\mathbf{G}_\mathcal{G}$ between a challenger and an adversary A. A is said to win if $\mathbf{G}_\mathcal{G}^\mathsf{A} = 1$; $\mathbf{CDH}_\mathcal{G}^\mathsf{A} = 1$ when $Z = x_1x_2G$. We define an advantage and a running time of an adversary A in $\mathbf{G}_\mathcal{G}$ as $\mathsf{Adv}_{\mathcal{G},\mathsf{A}}^\mathbf{G} := \Pr[\mathbf{G}_\mathcal{G}^\mathsf{A} = 1]$ and $\mathsf{Time}_{\mathcal{G},\mathsf{A}}^\mathbf{G}$, respectively.

Generic Group Model (GGM). In the GGM, an adversary A_gen is not given actual representations of group elements but the elements via abstract handles. For example, an adversary A_gen in a security game $\mathbf{CDH}_\mathcal{G}$ receives a group description $\mathcal{G} := (\mathbb{G}, \mathtt{00}, p)$ and $(\mathtt{01}, \mathtt{02})$ from a challenger. Here, $\mathbb{G}$ only contains an information of an additive cyclic group of a prime-order p. The adversary A_gen is able to perform group operations only via oracle queries, e.g., a generic adversary A_gen queries $(\mathtt{01}, \mathtt{02}, +)$ to an oracle and obtains $\mathtt{03}$, where $\mathtt{01} = X_1, \mathtt{02} = X_2$, and $\mathtt{03} = X_1 + X_2$. Since

a behavior of the generic adversary $\mathsf{A}_{\mathsf{gen}}$ is independent of actual group representations, it works in any groups.

Some computational problems that are hard in the GGM may not be hard when instantiated in concrete groups. However, the GGM is still useful since it enables us to obtain information theoretic lower bounds. We use a notion of (ε, t)-*hard* if for all generic algorithms $\mathsf{A}_{\mathsf{gen}}$ in a game $\mathbf{G}_{\mathcal{G}}$

$$\mathsf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{gen}}} \leq t \quad \Rightarrow \quad \mathsf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}_{\mathsf{gen}}} \leq \varepsilon$$

holds. The following fact is known for the discrete logarithm problem.

Lemma 1 (Generic Hardness of DL [Sho97, Mau05])**.** *The discrete logarithm problem is* $(t^2/p, t)$*-hard in the GGM.*

Algebraic Algorithm. Now, we review a notion of an *algebraic algorithm* defined by Fuchsbauer et al. [FKL18]. An algebraic algorithm is able to output group elements only via group additions of given elements. Furthermore, the algebraic algorithm should also output a *representation* which indicates how outputted group elements are calculated with respect to given elements.

Definition 14 (Algebraic Algorithm, Definition 2.1 of [FKL18])**.** *An algorithm* $\mathsf{A}_{\mathsf{alg}}$ *executed in an algebraic security game* $\mathbf{G}_{\mathcal{G}}$ *in a cyclic group* $\mathcal{G} := (\mathbb{G}, G, p)$ *is called* algebraic *if for all group elements* $Z \in \mathbb{G}$ *that* $\mathsf{A}_{\mathsf{alg}}$ *outputs, it additionally returns the representation of* Z *with respect to given group elements. Specifically, if* $\vec{X} := (X_0, \ldots, X_\ell) \in \mathbb{G}^{\ell+1}$*, where* $X_0 := G$*, is the list of group elements that* $\mathsf{A}_{\mathsf{alg}}$ *has received so far, then* $\mathsf{A}_{\mathsf{alg}}$ *must also return a vector* $\vec{z} := (z_i)_{0 \leq i \leq \ell} \in \mathbb{Z}_p^{\ell+1}$ *such that* $Z = \sum_{i=0}^{\ell} z_i X_i$*. We use* $[Z]_{\vec{z}}$ *to denote such an output.*

We remark that every generic algorithm $\mathsf{A}_{\mathsf{gen}}$ can be modeled as an algebraic one. A generic algorithm $\mathsf{A}_{\mathsf{gen}}$ is able to output only group elements which are derived from group additions of given elements as an algebraic algorithm. Furthermore, by keeping a record of all oracle queries, a generic algorithm $\mathsf{A}_{\mathsf{gen}}$ is able to output a group element Z along with its representation $\vec{z}$. Hence, a generic algorithm $\mathsf{A}_{\mathsf{gen}}$ is able to behave as an algebraic algorithm. Moreover, let $\mathsf{A}_{\mathsf{alg}}$ and $\mathsf{B}_{\mathsf{gen}}$ be an algebraic and a generic algorithm, respectively. Then, $\mathsf{B}_{\mathsf{alg}} := \mathsf{B}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ is also an algebraic algorithm.

Reduction between Algebraic Security Games. Let $\mathbf{G}_{\mathcal{G}}$ and $\mathbf{H}_{\mathcal{G}}$ be two algebraic security games. Please keep in mind that $\mathbf{G}_{\mathcal{G}}$ and $\mathbf{H}_{\mathcal{G}}$ will be the game for the CDH variants and the (B)DL, respectively. We use $\mathbf{H}_{\mathcal{G}} \Rightarrow_{\mathsf{alg}} \mathbf{G}_{\mathcal{G}}$ to denote an existence of a *generic* and tight reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ such that for every algorithm A, an algorithm $\mathsf{B} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{A}}$ satisfies

$$\mathsf{Adv}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}} = \mathsf{Adv}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}} \qquad \text{and} \qquad \mathsf{Time}^{\mathbf{H}}_{\mathcal{G},\mathsf{B}} = \mathsf{Time}^{\mathbf{G}}_{\mathcal{G},\mathsf{A}}.$$

The crucial point of the definition is that a reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ is generic. Hence, if $\mathsf{A} = \mathsf{A}_{\mathsf{alg}}$ is algebraic, $\mathsf{B} = \mathsf{B}_{\mathsf{alg}}$ is also algebraic. Furthermore, if $\mathsf{A} = \mathsf{A}_{\mathsf{gen}}$ is generic, $\mathsf{B} = \mathsf{B}_{\mathsf{gen}}$ is also generic. Thanks to the generic reduction algorithm $\mathsf{R}_{\mathsf{gen}}$, we are able to obtain information theoretic lower bounds of CDH variants as follows by combining with Lemma 1.

Lemma 2 (Lemma 2.2 of [FKL18])**.** *Let* $\mathbf{G}_{\mathcal{G}}$ *and* $\mathbf{H}_{\mathcal{G}}$ *be algebraic security games such that* $\mathbf{H}_{\mathcal{G}} \Rightarrow_{\mathsf{alg}} \mathbf{G}_{\mathcal{G}}$ *and winning* $\mathbf{H}_{\mathcal{G}}$ *is* (ε, t)*-hard in the GGM. Then,* $\mathbf{G}_{\mathcal{G}}$ *is* (ε, t)*-hard in the GGM.*

3 Reduction from DL in Cyclic Groups

In this section, we show several generic and tight reductions from the DL to the CDH variants in cyclic groups. We first provide a direct reduction to the CDH in Section 3.1. Then, we provide our master theorem in Section 3.2.

3.1 Basic Reduction: From DL to CDH

In this section, we show a basic approach of this paper by providing a generic and tight reduction from the DL to the CDH in the AGM.

Theorem 1. $\mathbf{DL}_{\mathcal{G}} \Rightarrow_{\mathsf{alg}} \mathbf{CDH}_{\mathcal{G}}$.

Proof. We construct a generic and tight reduction algorithm $\mathsf{R}_{\mathsf{gen}}$. Specifically, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $\mathbf{CDH}_{\mathcal{G}}$ only once and construct an algebraic adversary $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ on the $\mathbf{DL}_{\mathcal{G}}$.

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ is given a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and an instance of the $\mathbf{DL}_{\mathcal{G}}$, i.e., $X := xG \in \mathbb{G}$ for an unknown $x \in \mathbb{Z}_p$. Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ creates an instance of the $\mathbf{CDH}_{\mathcal{G}}$ as follows: Pick a random $r \xleftarrow{\$} \mathbb{Z}_p$ and compute

$$X_2 := X + rG = (x + r)G \in \mathbb{G},$$

then set

$$(X_1 := X, X_2) \in \mathbb{G}^2.$$

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ gives a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and group elements $(X_1, X_2) \in \mathbb{G}^2$ to $\mathsf{A}_{\mathsf{alg}}$. Observe that (X_1, X_2) is a valid CDH instance by implicitly setting

$$(x_1, x_2) = (x, x + r)$$

since x_2 is independently distributed of x_1 to uniform in $\mathbb{Z}_p$ from $\mathsf{A}_{\mathsf{alg}}$'s view. Hence, an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ outputs a correct solution $[Z]_{\vec{z}}$ with an advantage $\mathsf{Adv}^{\mathbf{CDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$ and a running time $\mathsf{Time}^{\mathbf{CDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$.

Next, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses $[Z]_{\vec{z}}$ outputted by an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $\mathbf{CDH}_{\mathcal{G}}$ and computes a solution of the $\mathbf{DL}_{\mathcal{G}}$. Assume the output is a correct solution of the CDH, i.e., $Z = x_1 x_2 G$. It holds with probability $\mathsf{Adv}^{\mathbf{CDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$. Then, the representation vector $\vec{z} := (z_0, z_1, z_2)$ satisfies

$$\begin{aligned} x_1 x_2 G = x(x + r)G &= z_0 G + z_1 X + z_2 Y \\ &= (z_0 + z_1 x + z_2(x + r))G. \end{aligned}$$

Hence, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ obtains the following univariate equation modulo a prime p:

$$\begin{aligned} & x(x + r) = z_0 + z_1 x + z_2(x + r) \mod p \\ \Leftrightarrow\ & x^2 + (r - z_1 - z_2)x - z_0 - z_2 r = 0 \mod p. \end{aligned}$$

Observe that the left hand side is a degree 2 monic polynomial; hence, a non-zero polynomial. Since the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ knows a value of r, it is able to find all solutions for x in polynomial time. By checking $xG = X$, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ successfully finds a correct solution of the $\mathbf{DL}_{\mathcal{G}}$. □

Table 1: Applicability of our technique in cyclic groups

problem	$(f_i)_{i\in[\ell]}$	g	$\max \deg f_i$	$\deg g$	reduction?
CDH	(x_1, x_2)	x_1x_2	1	2	Yes
k-PDH	$(x_i)_{i\in[k]}$	$x_1 \cdots x_k$	1	k	Yes
k-EDH	x	x^k	1	k	Yes
k-Lin$^{(1)}$	$\left((x_i)_{i\in[k]}, (x_iy_i)_{i\in[k]}\right)$	$\sum_{i=1}^{k} y_i$	1	1	No
k-Lin$^{(2)}$	$\left((x_i)_{i\in[k]}, (x_iy_i)_{i\in[k-1]}, \sum_{i=1}^{k} y_i\right)$	x_ky_k	1	2	Yes

By combining with Lemmas 1, 2, and Theorem 1, we are able to obtain an information theoretic lower bound for the CDH.

Theorem 2 (Generic Hardness of CDH)**.** *The computational Diffie-Hellman problem in Definition 2 is $(t^2/p, t)$-hard in the generic group model.*

3.2 Master Theorem in Cyclic Groups

In this subsection, we provide the following master theorem in cyclic groups to indicate the power of our technique.

Theorem 3 (Master Theorem in Cyclic Groups)**.** $\mathbf{DL}_{\mathcal{G}} \Rightarrow_{\mathsf{alg}} \mathbf{GDH}_{\mathcal{G}}$ *holds when the following conditions hold:*

(1) $\deg_{x_1,\ldots,x_m} f_i(x_1, \ldots, x_m, y_1, \ldots, y_n) \in \{0, 1\}$ *for all* $i \in [\ell]$,

(2) $\deg g(x_1, \ldots, x_m) > 1$.

Before providing a proof, we summarize the CDH variants which we studied in Section 3 and the conditions of Theorem 3 in Table 1. As the table shows, CDH, k-PDH, k-EDH, and k-Lin$^{(2)}$ satisfy the conditions (1) and (2) in Theorem 3. Hence, as immediate corollary of the master theorem, we are able to provide generic and tight reductions from the DL to the k-PDH, k-EDH, and k-Lin$^{(2)}$. Unfortunately, the k-Lin$^{(1)}$ does not satisfy the condition (2). Hence, we will provide a tailor-made reduction for the k-Lin$^{(1)}$ later.

Then, we show a proof of Theorem 3. In advance, we claim that the condition (1) will be used to ensure that the reduction algorithm is able to produce all group elements of the GDH during a reduction, while both the conditions (1) and (2) will be used to ensure that the modular equation never becomes a zero polynomial.

Proof. We construct a generic and tight reduction algorithm $\mathsf{R}_{\mathsf{gen}}$. Specifically, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $\mathbf{GDH}_{\mathcal{G}}$ only once and construct an algebraic adversary $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ on the $\mathbf{DL}_{\mathcal{G}}$.

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ is given a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and an instance of the $\mathbf{DL}_{\mathcal{G}}$, i.e., $X := xG \in \mathbb{G}$ for an unknown $x \in \mathbb{Z}_p$. Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ creates an instance of the $\mathbf{GDH}_{\mathcal{G}}$ as follows: Pick random $(r_2, \ldots, r_m, s_1, \ldots, s_n) \xleftarrow{\$} \mathbb{Z}_p^{m+n-1}$ and compute

$$X_i := f_i(x_1, \ldots, x_m, y_1, \ldots, y_n)G \in \mathbb{G}$$

for all $i \in [\ell]$ by implicitly setting

$$(x_1, x_2, \ldots, x_m, y_1, \ldots, y_n) = (x, x + r_2, \ldots, x + r_m, s_1, \ldots, s_n).$$

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ is able to compute all the group elements thanks to the condition (1). Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ gives a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and group elements $(X_1, \ldots, X_\ell) \in \mathbb{G}^\ell$ to $\mathsf{A}_{\mathsf{alg}}$. Observe that $(X_1, \ldots, X_\ell)$ is a valid GDH instance since $(x_2, \ldots, x_m)$ is independently distributed of x_1 to uniform in $\mathbb{Z}_p^{m-1}$ from $\mathsf{A}_{\mathsf{alg}}$'s view. Hence, an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ outputs a correct solution $[Z]_{\vec{z}}$ with an advantage $\mathsf{Adv}^{\mathbf{GDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$ and a running time $\mathsf{Time}^{\mathbf{GDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$.

Next, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses $[Z]_{\vec{z}}$ outputted by an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $\mathbf{GDH}_{\mathcal{G}}$ and computes a solution of the $\mathbf{DL}_{\mathcal{G}}$. Assume the output is a correct solution of the GDH, i.e., $Z = g(x_1, \ldots, x_m)G$. It holds with probability $\mathsf{Adv}^{\mathbf{GDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$. Then, the representation vector $\vec{z} := (z_0, z_1, \ldots, z_\ell)$ satisfies

$$\begin{aligned} & g(x_1, \ldots, x_m)G \\ &= z_0 G + z_1 X_1 + \cdots + z_\ell X_\ell \\ &= (z_0 + z_1 f_1(x_1, \ldots, x_m, y_1, \ldots, y_n) + \cdots + z_\ell f_\ell(x_1, \ldots, x_m, y_1, \ldots, y_n))G. \end{aligned}$$

Hence, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ obtains the following univariate equation modulo a prime p:

$$\begin{aligned} & g(x, x + r_2, \ldots, x + r_m) \\ &= z_0 + \sum_{i=1}^{\ell} z_i f_i(x, x + r_2, \ldots, x + r_m, s_1, \ldots, s_n) \mod p. \end{aligned}$$

Observe that a degree of the left and the right hand side with respect to a variable x is strictly larger than 1 and exactly 1 respectively due to the conditions (1) and (2). Hence, the modular equation never becomes a zero polynomial. Since the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ knows values of $(r_2, \ldots, r_m, s_1, \ldots, s_n)$, it is able to find all solutions for x in polynomial time. By checking $xG = X$, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ successfully finds a correct solution of the $\mathbf{DL}_{\mathcal{G}}$. □

By combining with Lemmas 1, 2, and Theorem 3, we are able to obtain an information theoretic lower bound for the GDH as follows.

Theorem 4 (Generic Hardness of GDH)**.** *The generalized Diffie-Hellman problem in Definition 9 is $(t^2/p, t)$-hard in the generic group model.*

4 Reduction from BDL in Bilinear Groups

In this section, we show tight reductions from the bilinear discrete logarithm problem to the bilinear Diffie-Hellman problem in an *algebraic bilinear group model* which we define in Section 4.1. In Section 4.2, we provide a reduction to the BDH. Finally, we provide our master theorem in Section 4.3.

4.1 Algebraic Bilinear Group Model

In advance of the reduction, we define an algebraic bilinear algorithm. The definition is analogous to Definition 14 in the sense that the algebraic bilinear algorithm is able to output only group elements which are derived from group additions in $\mathbb{G}$, group multiplications in $\mathbb{G}_T$, and pairing e of given

elements. Furthermore, the algebraic bilinear algorithm should also output a *representation* which indicates how outputted group elements are calculated. In this paper, we study computational problems in bilinear groups whose solutions Z are elements in $\mathbb{G}_T$. Hence, we define a representation so that it records how Z is computed by group multiplications of given elements in $\mathbb{G}_T$ and pairing of given elements in $\mathbb{G}$. We formally provide a definition as follows.

Definition 15 (Algebraic Bilinear Algorithm)**.** *An algorithm* $\mathsf{A}_{\mathsf{alg}}$ *executed in an algebraic security game* $\mathbf{G}_{\mathcal{G}}$ *for* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ *is called* algebraic *if for all group elements* $Z \in \mathbb{G}_T$ *that* $\mathsf{A}_{\mathsf{alg}}$ *outputs, it additionally return the representation of* Z *with respect to given group elements. Specifically, if* $\vec{X} := (X_0, \ldots, X_k) \in \mathbb{G}^{k+1}$*, where* $X_0 := G$*, and* $\vec{Y} := (Y_1, \ldots, Y_\ell) \in \mathbb{G}_T^\ell$ *are the list of group elements that* $\mathsf{A}_{\mathsf{alg}}$ *has received so far, then* $\mathsf{A}_{\mathsf{alg}}$ *must also return a vector* $\vec{z} := ((z_{ij})_{0\le i\le j\le k}, (z'_i)_{1\le i\le \ell}) \in \mathbb{Z}_p^{\frac{(k+1)(k+2)}{2}+\ell}$ *such that* $Z = \left(\prod_{0\le i\le j\le k} e(X_i, X_j)^{z_{ij}}\right) \cdot \left(\prod_{i=1}^{\ell} Y_i^{z'_i}\right)$*. We denote such an output as* $[Z]_{\vec{z}}$*.*

We note that the BDH and the ℓ-wBDI does not take elements in $\mathbb{G}_T$ as the input. Therefore, the algorithm outputs Z along with a vector $\vec{z} \in \mathbb{Z}_p^{\frac{(k+1)(k+2)}{2}}$.

4.2 From BDL to BDH

In this subsection, we extend the approach in Section 3 and prove the following reduction.

Theorem 5. $\mathbf{BDL}_{\mathcal{G}} \Rightarrow_{\mathsf{alg}} \mathbf{BDH}_{\mathcal{G}}$.

Proof. We construct a generic and tight reduction algorithm $\mathsf{R}_{\mathsf{gen}}$. Specifically, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $\mathbf{BDH}_{\mathcal{G}}$ only once and construct an algebraic adversary $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ on the $\mathbf{BDL}_{\mathcal{G}}$.

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ is given a bilinear group description $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ and an instance of the $\mathbf{BDL}_{\mathcal{G}}$, i.e., $X := xG \in \mathbb{G}$ for an unknown $x \in \mathbb{Z}_p$. Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ creates an instance of the $\mathbf{BDH}_{\mathcal{G}}$ as follows: Pick a random $(r, s) \xleftarrow{\$} \mathbb{Z}_p^2$ and compute

$$(X_2 := X + rG = (x + r)G, \quad X_3 := X + sG = (x + s)G) \in \mathbb{G}^2,$$

then set

$$(X_1 := X, X_2, X_3) \in \mathbb{G}^3.$$

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ gives a bilinear group description $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ and group elements $(X_1, X_2, X_3) \in \mathbb{G}^3$ to $\mathsf{A}_{\mathsf{alg}}$. Observe that (X_1, X_2, X_3) is a valid BDH instance since (x_2, x_3) is independently distributed of x to uniform in $\mathbb{Z}_p^2$ from $\mathsf{A}_{\mathsf{alg}}$'s view. Hence, an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ outputs a correct solution $[Z]_{\vec{z}}$ with an advantage $\mathsf{Adv}^{\mathbf{BDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$ and a running time $\mathsf{Time}^{\mathbf{BDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$.

Next, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses $[Z]_{\vec{z}}$ outputted by an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $\mathbf{BDH}_{\mathcal{G}}$ and computes a solution of the $\mathbf{BDL}_{\mathcal{G}}$. Assume the output is a correct solution of the BDH, i.e., $Z = e(G, G)^{x_1x_2x_3}$. It holds with probability $\mathsf{Adv}^{\mathbf{BDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$. Then, we use $X_0 := G$ for notational convenience and the representation vector $\vec{z} := (z_{ij})_{0\le i\le j\le 3}$ satisfies

$$\begin{aligned} & e(G, G)^{x_1x_2x_3} \\ &= e(G, G)^{x(x+r)(x+s)} \\ &= \prod_{0\le i\le j\le 3} e(X_i, X_j)^{z_{ij}} \end{aligned}$$

$$= e(G,G)^{z_{00}+z_{01}x+z_{02}(x+r)+z_{03}(x+s)+z_{11}x^2+z_{12}x(x+r)+z_{13}x(x+s)+z_{22}(x+r)^2+z_{23}(x+r)(x+s)+z_{33}(x+s)^2}.$$

Hence, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ obtains the following univariate equation modulo a prime p:

$$\begin{aligned}
& x(x+r)(x+s) \\
=\ & z_{00} + z_{01}x + z_{02}(x+r) + z_{03}(x+s) + z_{11}x^2 + z_{12}x(x+r) \\
& + z_{13}x(x+s) + z_{22}(x+r)^2 + z_{23}(x+r)(x+s) + z_{33}(x+s)^2 \mod p \\
\Leftrightarrow\ & x^3 + (r+s-z_{11}-z_{12}-z_{13}-z_{22}-z_{23}-z_{33})x^2 \\
& + (rs - z_{01} - z_{02} - z_{03} - rz_{12} - sz_{13} - 2rz_{22} - (r+s)z_{23} - 2sz_{33})x \\
& - z_{00} - rz_{02} - sz_{03} - r^2 z_{22} - rsz_{23} - s^2 z_{33} = 0 \mod p.
\end{aligned}$$

Observe that the left hand side is a degree 3 monic polynomial; hence, a non-zero polynomial. Since the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ knows values of r and s, it is able to find all solutions for x in polynomial time. By checking $xG = X$, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ successfully finds a correct solution of the $\mathbf{BDL}_{\mathcal{G}}$. □

By combining with Lemmas 1, 2, and Theorem 5, we are able to obtain an information theoretic lower bound for the BDH.

Theorem 6 (Generic Hardness of BDH)**.** *The bilinear Diffie-Hellman problem in Definition 11 is $(t^2/p, t)$-hard in the generic group model.*

4.3 Master Theorem in Bilinear Groups

In this subsection, we provide the following master theorem in bilinear groups to indicate the power of our technique.

Theorem 7 (Master Theorem in Bilinear Groups)**.** $\mathbf{BDL}_{\mathcal{G}} \Rightarrow_{\mathsf{alg}} \mathbf{GBDH}_{\mathcal{G}}$ *holds when the following conditions hold:*

(1) $\deg_{x_1,\ldots,x_m} f_i(x_1,\ldots,x_m,y_1,\ldots,y_n) \in \{0,1\}$ *for all* $i \in [k]$,

(2) $\deg_{x_1,\ldots,x_m} g_i(x_1,\ldots,x_m,y_1,\ldots,y_n) \in \{0,1,2\}$ *for all* $i \in [\ell]$,

(3) $\deg h(x_1,\ldots,x_m) > 2$.

Before providing a proof, we summarize the BDH and the ℓ-wBDHI which we studied in Section 4 and the conditions of Theorem 7 in Table 2. Since these problems do not take group elements in $\mathbb{G}_T$ as the input, we omit the condition (2) in the table. As the table shows, the BDH which we provided reductions satisfies the conditions (1) and (3) in Theorem 7. However, the ℓ-wBDHI does not satisfy the condition (1) for $\ell > 1$. Hence, as immediate corollary of the master theorem, we are able to provide generic and tight reductions from the BDL to the 1-wBDHI. In this paper, we are not able to provide a tailor-made reduction for ℓ-wBDHI for $\ell > 1$.

Then, we show a proof of Theorem 7. In advance, we claim that the conditions (1) and (2) will be used to ensure that the reduction algorithm is able to produce all group elements of the GBDH during a reduction, while all the conditions (1), (2), and (3) will be used to ensure that the modular equation never becomes a zero polynomial.

Table 2: Applicability of our technique in bilinear groups

problem	$(f_i)_{i\in[k]}$	h	$\max\deg f_i$	$\deg h$	reduction?
BDH	$(x_i)_{i\in[3]}$	$x_1x_2x_3$	1	3	Yes
ℓ-wBDHI	$((x_1^i)_{i\in[\ell]}, x_2)$	$x_1^{\ell+1}x_2$	ℓ	$\ell+2$	$\ell=1$

Proof. We construct a generic and tight reduction algorithm $\mathsf{R}_{\mathsf{gen}}$. Specifically, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $\mathbf{GBDH}_{\mathcal{G}}$ only once and construct an algebraic adversary $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ on the $\mathbf{BDL}_{\mathcal{G}}$.

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ is given a bilinear group description $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ and an instance of the $\mathbf{BDL}_{\mathcal{G}}$, i.e., $X := xG \in \mathbb{G}$ for an unknown $x \in \mathbb{Z}_p$. Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ creates an instance of the $\mathbf{GBDH}_{\mathcal{G}}$ as follows: Pick random $(r_2, \ldots, r_m, s_1, \ldots, s_n) \xleftarrow{\$} \mathbb{Z}_p^{m+n-1}$ and compute

$$X_i := f_i(x_1, \ldots, x_m, y_1, \ldots, y_n)G \in \mathbb{G}$$

for all $i \in [k]$ and

$$Y_j := g_j(x_1, \ldots, x_m, y_1, \ldots, y_n)G \in \mathbb{G}_T$$

for all $j \in [\ell]$ by implicitly setting

$$(x_1, x_2, \ldots, x_m, y_1, \ldots, y_n) = (x, x + r_2, \ldots, x + r_m, s_1, \ldots, s_n).$$

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ is able to compute all the group elements thanks to the conditions (1) and (2). Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ gives a bilinear group description $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ and group elements $(X_1, \ldots, X_k, Y_1, \ldots, Y_\ell) \in \mathbb{G}^k \times \mathbb{G}_T^\ell$ to $\mathsf{A}_{\mathsf{alg}}$. Observe that $(X_1, \ldots, X_k, Y_1, \ldots, Y_\ell)$ is a valid GBDH instance since $(x_2, \ldots, x_m)$ is independently distributed of x_1 to uniform in $\mathbb{Z}_p^{m-1}$ from $\mathsf{A}_{\mathsf{alg}}$'s view. Hence, an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ outputs a correct solution $[Z]_{\vec{z}}$ with an advantage $\mathsf{Adv}^{\mathbf{GBDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$ and a running time $\mathsf{Time}^{\mathbf{GBDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$.

Next, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses $[Z]_{\vec{z}}$ outputted by an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $\mathbf{GBDH}_{\mathcal{G}}$ and computes a solution of the $\mathbf{BDL}_{\mathcal{G}}$. Assume the output is a correct solution of the GBDH, i.e., $Z = h(x_1, \ldots, x_m, y_1, \ldots, y_n)G$. It holds with probability $\mathsf{Adv}^{\mathbf{GBDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$. Then, the representation vector $\vec{z} := (z_0, (z_{ij})_{0\le i\le j\le k}, (z'_i)_{1\le i\le \ell})$ satisfies

$$\begin{aligned}
& e(G,G)^{h(x_1,\ldots,x_m,y_1,\ldots,y_n)} \\
&= \left(\prod_{0\le i\le j\le k} e(X_i, X_j)^{z_{ij}}\right) \cdot \left(\prod_{1\le i\le \ell} Y_i^{z'_i}\right) \\
&= e(G,G)^{\sum_{0\le i\le j\le k} z_{ij} f_i(x_1,\ldots,x_m,y_1,\ldots,y_n)\cdot f_j(x_1,\ldots,x_m,y_1,\ldots,y_n) + \sum_{i=1}^{\ell} z'_i g_i(x_1,\ldots,x_m,y_1,\ldots,y_n)}.
\end{aligned}$$

Hence, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ obtains the following univariate equation modulo a prime p:

$$\begin{aligned}
& h(x, x + r_2, \ldots, x + r_m, s_1, \ldots, s_n) \\
&= \sum_{0\le i\le j\le k} z_{ij} f_i(x, x + r_2, \ldots, x + r_m, s_1, \ldots, s_n) \cdot f_j(x, x + r_2, \ldots, x + r_m, s_1, \ldots, s_n) \\
&\quad + \sum_{i=1}^{\ell} z'_i g_i(x, x + r_2, \ldots, x + r_m, s_1, \ldots, s_n) \mod p.
\end{aligned}$$

Observe that a degree of the left and the right hand side with respect to a variable x is strictly larger than 2 and at most 2 respectively due to the conditions (1), (2), and (3). Hence, the modular equation never becomes a zero polynomial. Since the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ knows values of $(r_2, \ldots, r_m, s_1, \ldots, s_n)$, it is able to find all solutions for x in polynomial time. By checking $xG = X$, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ successfully finds a correct solution of the $\mathbf{BDL}_{\mathcal{G}}$. □

By combining with Lemmas 1, 2, and Theorem 7, we are able to obtain an information theoretic lower bound for the GBDH as follows.

Theorem 8 (Generic Hardness of GBDH)**.** *The generalized bilinear Diffie-Hellman problem in Definition 13 is $(t^2/p, t)$-hard in the generic group model.*

5 DL to k-Lin Reduction

In this section, we provide a generic and tight reduction from the DL to the k-Lin$^{(1)}$ in an ad-hoc manner.

As described in Section 3.2, our master theorem (Theorem 3) does not capture the k-Lin$^{(1)}$ problem. The core trick of the above reduction for the CDH consists of the following two steps:

- embedding the DL solution x into group elements of the CDH instance (X_1, X_2),
- constructing a modular equation with a *non-zero* polynomial whose solution is x.

In particular, by following the same approach, we are not able to ensure that a modular equation becomes non-zero. For example, for simplicity we explain how the attempt fails for the 1-Lin$^{(1)}$. To provide a reduction from the DL to the 1-Lin$^{(1)}$ in the same way, we try to embed the DL solution x into x_1 *and/or* y_1 of the 1-Lin$^{(1)}$ instance. We note that the DL solution x cannot be embedded into x_1 *and* y_1, simultaneously. In particular, the reduction algorithm is not able to create group elements whose discrete logarithm have a term of x of degree 2, e.g., x^2G. The intractability readily follows from the fact that the square DH is computationally equivalent to the DL.

Hence, we try to continue the approach by embedding the DL solution x into x_1 *or* y_1. When we embed x into x_1 and create a 1-Lin$^{(1)}$ instance $(X_1 = X, Y_1 = cX)$ by implicitly setting $(x_1, y_1) = (x, c)$ with a random $c \in \mathbb{Z}_p$, the 1-Lin$^{(1)}$ algorithm outputs $[Z]_{\vec{z}}$ such that $Z = z_0G + z_1X_1 + z_2Y_1$. When Z is a correct solution $Z = y_1G$,

$$\begin{aligned} y_1G = cG &= z_0G + z_1X_1 + z_2Y_1 \\ &= (z_0 + z_1x + z_2cx)\,G \end{aligned}$$

holds and the reduction algorithm obtains a modular equation

$$\begin{aligned} & c = z_0 + z_1x + z_2cx \mod p \\ \Leftrightarrow\ & (z_1 + z_2c)x + z_0 - c = 0 \mod p. \end{aligned}$$

Observe that the left hand side is a zero polynomial when $z_1 + z_2c = z_0 - c = 0$. Similarly, when we embed x into y_1 and create a 1-Lin$^{(1)}$ instance $(X_1 = cG, Y_1 = cX)$ by implicitly setting $(x_1, y_1) = (c, x)$ with a random $c \in \mathbb{Z}_p$,

$$\begin{aligned} y_1G = xG &= z_0G + z_1X_1 + z_2Y_1 \\ &= (z_0 + z_1c + z_2cx)\,G \end{aligned}$$

holds and the reduction algorithm obtains a modular equation

$$\begin{aligned} & x = z_0 + z_1 c + z_2 cx \mod p \\ \Leftrightarrow\ & (z_2 c - 1)x + z_0 + z_1 c = 0 \mod p. \end{aligned}$$

Observe that the left hand side is a zero polynomial when $z_2 c - 1 = z_0 + z_1 c = 0$. Hence, the reduction algorithm may fail even when the 1-Lin$^{(1)}$ algorithm outputs a correct solution $Z = y_1 G$.

The reduction of the CDH (and our master theorem) avoids the problem by using the fact that the x_1 and x_2 which are the discrete logarithms of the input (X_1, X_2) are linear in x while $x_1 x_2$ which is the discrete logarithm of the CDH solution Z is quadratic in x. In other words, the resulting modular equation has to become a monic *non-zero* polynomial.

However, by modifying the approach, we are still able to provide a generic and tight reduction from the DL to the k-Lin$^{(1)}$.

Theorem 9. $\mathbf{DL}_{\mathcal{G}} \Rightarrow_{\mathsf{alg}} k\text{-}\mathbf{Lin}^{(1)}_{\mathcal{G}}$.

Here, for simplicity we prove $\mathbf{DL}_{\mathcal{G}} \Rightarrow_{\mathsf{alg}} 1\text{-}\mathbf{Lin}^{(1)}_{\mathcal{G}}$. Note that there is a reduction from the $1\text{-}\mathbf{Lin}^{(1)}_{\mathcal{G}}$ to the $k\text{-}\mathbf{Lin}^{(1)}_{\mathcal{G}}$ in the standard model. Hence, the proof for $1\text{-}\mathbf{Lin}^{(1)}_{\mathcal{G}}$ is sufficient to prove Theorem 9.

Proof. We construct a generic and tight reduction algorithm $\mathsf{R}_{\mathsf{gen}}$. Specifically, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $1\text{-}\mathbf{Lin}^{(1)}_{\mathcal{G}}$ only once and construct an algebraic adversary $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}^{\mathsf{A}_{\mathsf{alg}}}_{\mathsf{gen}}$ on the $\mathbf{DL}_{\mathcal{G}}$.

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ is given a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and an instance of the $\mathbf{DL}_{\mathcal{G}}$, i.e., $X := xG \in \mathbb{G}$ for an unknown $x \in \mathbb{Z}_p$. Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ creates an instance of the $1\text{-}\mathbf{Lin}^{(1)}_{\mathcal{G}}$ as follows: Pick random $c \xleftarrow{\$} \mathbb{Z}_p$ and compute

$$Y_1 := cG \in \mathbb{G},$$

then set

$$(X_1 := X, Y_1) \in \mathbb{G}^2.$$

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ gives a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and group elements $(X_1, Y_1) \in \mathbb{G}^2$ to $\mathsf{A}_{\mathsf{alg}}$. Observe that (X_1, Y_1) is a valid 1-Lin$^{(1)}$ instance by implicitly setting

$$(x_1, y_1) = (x, c/x)$$

since y_1 is independently distributed of x_1 to uniform in $\mathbb{Z}_p$ from $\mathsf{A}_{\mathsf{alg}}$'s view. Hence, an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ outputs a correct solution $[Z]_{\vec{z}}$ with an advantage $\mathsf{Adv}^{1\text{-}\mathbf{Lin}^{(1)}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$ and a running time $\mathsf{Time}^{1\text{-}\mathbf{Lin}^{(1)}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$.

Next, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses $[Z]_{\vec{z}}$ outputted by an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $1\text{-}\mathbf{Lin}^{(1)}_{\mathcal{G}}$ and computes a solution of the $\mathbf{DL}_{\mathcal{G}}$. Assume the output is a correct solution of the 1-Lin$^{(1)}$, i.e., $Z = y_1 G$. It holds with probability $\mathsf{Adv}^{\mathbf{1\text{-}Lin}^{(1)}_{\mathcal{G}}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$. Then, the representation vector $\vec{z} := (z_0, z_1, z_2)$ satisfies

$$\begin{aligned} y_1 G = (c/x)G &= z_0 G + z_1 X_1 + z_2 Y_1 \\ &= (z_0 + z_1 x + z_2 c)\, G \end{aligned}$$

Hence, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ obtains the following univariate equation modulo a prime p:

$$\begin{aligned} c/x &= z_0 + z_1 x + z_2 c \mod p \\ \Leftrightarrow z_1 x^2 + (z_0 + z_2 c)x - c &= 0 \mod p. \end{aligned}$$

Observe that the left hand side is not a monic polynomial. However, it has to be a non-zero polynomial due to the constant term c. Since the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ knows values of c, it is able to find all solutions for x in polynomial time. By checking $xG = X$, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ successfully finds a correct solution of the $\mathbf{DL}_{\mathcal{G}}$. □

By combining with Lemmas 1, 2, and Theorem 9, we are able to obtain an information theoretic lower bound for the k-Lin$^{(2)}$.

Theorem 10 (Generic Hardness of k-Lin$^{(2)}$)**.** *The computational k-linear problem in Definition 8 is $(t^2/p, t)$-hard in the generic group model.*

The DL to k-Lin$^{(1)}$ reduction implies that our master theorem is not perfect in the sense that there are other ways to provide reductions from the DL to the computational problems.

6 Conclusion

In this paper, we revisited the AGM which Fuchsbauer, Kiltz, and Loss [FKL18] gave a simple and clean definition to study the computational hardness of the CDH family. The AGM allows us to study the problem based on very simple arguments. Among their several results, we focused on the generic and tight reduction from the DL to the CDH. For the purpose, they used the square DH as the intermediate step. On the other hand, we provided the direct reduction from the DL to the CDH. We extended the approach and provided several reductions from the DL to the CDH variants in cyclic groups. By extending the definition of the AGM, we also studied the computational hardness of the BDH in the same way. Our approach was able to provide these reduction based on as simple arguments as Fuchsbauer et al.'s one. What is more, we formalized master theorems to indicate that to what kinds of computational problems can be reduced from the (B)DL by following our approach.

The additional contents of this paper may be more valuable. We claimed the limit of our master theorems by showing that we were not able to provide reductions for the standard computational variant of the k-Lin and the ℓ-wBDHI for $\ell > 1$ in the same way. We slightly modified our approach and provided a generic and tight reduction from the k-Lin to the DL in an ad-hoc manner. On the other hand, we were not able to provide such reductions for the ℓ-wBDHI.

Studying the CDH variants that were not studied in this paper is an arguably interesting topic (possibly variants which are not captured by our master theorems). One interesting open problem is formalizing a new master theorem to capture the reduction from the k-Lin to the DL simultaneously. Throughout this paper, we focused only on tight reductions so that the approach becomes as simple as possible. As opposed to our work, studying the computational hardness of CDH variants by allowing reasonable reduction loss should also be an interesting approach. The most important future directions of this work are extending the technique to *composite*-order groups and/or *decisional* problems.

Acknowledgement. We would like to thank anonymous reviewers of CT-RSA 2019 for their helpful comments and suggestions. This research was supported by JST CREST Grant Number JPMJCR14D6, Japan.

References

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[BDS98] Mike Burmester, Yvo Desmedt, and Jennifer Seberry. Equitable key escrow with limited time span (or, how to enforce time expiration cryptographically). In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security, Proceedings*, volume 1514 of *Lecture Notes in Computer Science*, pages 380–391. Springer, 1998.

[BDZ03] Feng Bao, Robert H. Deng, and Huafei Zhu. Variations of diffie-hellman problem. In Sihan Qing, Dieter Gollmann, and Jianying Zhou, editors, *Information and Communications Security, 5th International Conference, ICICS 2003, Proceedings*, volume 2836 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2003.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[Bis08] Gautam Biswas. Diffie-Hellman technique: extended to multiple two-party keys and one multi-party key. *IET Information Security*, 2(1):12–18, 2008.

[BL96] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 1996.

[Boy08] Xavier Boyen. The uber-assumption family. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing-Based Cryptography - Pairing 2008, Second International Conference, Proceedings*, volume 5209 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2008.

[BV98] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Proceedings*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71. Springer, 1998.

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

[EHK+17] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Luis Villar. An algebraic framework for Diffie-Hellman assumptions. *J. Cryptology*, 30(1):242–288, 2017.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology*

- *CRYPTO 2018 - 38th Annual International Cryptology Conference, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62. Springer, 2018.

[Gor93] Daniel M. Gordon. Discrete logarithms in *GF(P)* using the number field sieve. *SIAM J. Discrete Math.*, 6(1):124–138, 1993.

[Jou04] Antoine Joux. A one round protocol for tripartite diffie-hellman. *J. Cryptology*, 17(4):263–276, 2004.

[JS13] Tibor Jager and Jörg Schwenk. On the analysis of cryptographic assumptions in the generic ring model. *J. Cryptology*, 26(2):225–245, 2013.

[KMS04] Chisato Konoma, Masahiro Mambo, and Hiroki Shizuya. Complexity analysis of the cryptographic primitive problems through square-root exponent. *IEICE Transactions*, 87-A(5):1083–1091, 2004.

[KSW13] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *J. Cryptology*, 26(2):191–224, 2013.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix diffie-hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 729–758, 2016.

[MW96] Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 268–282. Springer, 1996.

[MW98] Ueli M. Maurer and Stefan Wolf. Lower bounds on generic algorithms in groups. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Proceedings*, volume 1403 of *Lecture Notes in Computer Science*, pages 72–84. Springer, 1998.

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

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[PH78] Stephen C. Pohlig and Martin E. Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.). *IEEE Trans. Information Theory*, 24(1):106–110, 1978.

[Pol78] J.M. Pollard. Monte carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978.

[PV05] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2005.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Proceedings*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.