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

Taiga Mizuide[†], Atsushi Takayasu[‡], Tsuyoshi Takagi[§]

June 29, 2019

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 showed 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 in the AGM, then used the known equivalence between the square Diffie-Hellman assumption and the CDH assumption in the standard model. 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 several 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. In this full version, we provide further applications (including the *matrix computational Diffie-Hellman assumption* and the *kernel matrix Diffie-Hellman assumption*) and a detailed overview of our techniques.

[*]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. This is the full version of [MTT19].

[†]Department of Creative Informatics, The University of Tokyo.

[‡]Department of Mathematical Informatics, The University of Tokyo and National Institute of Advanced Industrial Science and Technology (AIST). (`takayasu@mist.i.u-tokyo.ac.jp`)

[§]Department of Mathematical Informatics, The University of Tokyo.

Contents

1 Introduction **3**
- 1.1 Background . . . 3
- 1.2 Our Contributions . . . 4
- 1.3 Technical Overview . . . 5
- 1.4 Organization . . . 6

2 Computational Problems **6**
- 2.1 Diffie-Hellman Variants in Cyclic Groups . . . 6
- 2.2 Diffie-Hellman Variants in Symmetric Bilinear Groups . . . 7
- 2.3 Matrix Diffie-Hellman Problem . . . 8

3 Algebraic Group Model **11**

4 Reductions for Diffie-Hellman Variants in Cyclic Groups **13**
- 4.1 DL to CDH Reduction via Affine Embedding . . . 13
- 4.2 Master Theorem in Cyclic Groups . . . 14

5 Reductions for Diffie-Hellman Variants in Symmetric Bilinear Groups **16**
- 5.1 Algebraic Symmetric Bilinear Group Model . . . 16
- 5.2 BDL to CBDH Reduction via Affine Embedding . . . 16
- 5.3 Master Theorem in Symmetric Bilinear Groups . . . 18

6 Reduction for Matrix Computational Diffie-Hellman Problem **20**
- 6.1 Algebraic Group Model for Matrix Computational Diffie-Hellman Problem . . . 21
- 6.2 BDL to Computational k-Lin Reduction via Implicit Embedding . . . 21
- 6.3 Master Theorem for Matrix Computational Diffie-Hellman Problem . . . 22

7 Reduction for Matrix Kernel Diffie-Hellman Problem **26**
- 7.1 Algebraic Group Model for Matrix Kernel Diffie-Hellman Problem . . . 27
- 7.2 Master Theorem for Matrix Kernel Diffie-Hellman Problem . . . 27

8 Conclusion **29**

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, thanks 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 analogous 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

$$X_1 := X = xG \quad \text{and} \quad X_2 := X + rG = (x + r)G.$$

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 simplicity 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 computational 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.

In the preliminary version [MTT19], our master theorem in cyclic groups does not capture the computational k-linear problem. Thus, we gave a tailor-made reduction for the problem. In this full version, we provide a new master theorem for the matrix computational Diffie-Hellman problem that includes the computational k-linear problem as a special case. Furthermore, we show a new application called the kernel matrix Diffie-Hellman problem in this full version.

1.3 Technical Overview

In this subsection, we provide more detailed explanation of our technique, where the discussion did not appear in the preliminary version [MTT19].

In [FKL18], the reduction algorithm of Fuchsbauer et al. gives $(\mathcal{G}, X)$, which is exactly the DL instance, to a square DH algorithm. We call the reduction approach an *identity embedding* since the DL solution x is used only by $xG = X$ as a group element. In other words, even if we want to use an algorithm which takes multiple group elements $(X_1, X_2, \ldots)$ as the input, the identity embedding can embed the DL solution x into only one group element. Since the square DH is computationally equivalent to the CDH in the standard model, the identity embedding is enable to show a computational equivalence between the DL and the CDH by using the square DH as the intermediate bridge.

However, the identity embedding looks insufficient for providing a direct reduction from the DL to the CDH. The limitation of the identity embedding is that the DL solution is used only for one group element. When we try to use the identity embedding to provide a reduction for the CDH, the reduction algorithm randomly samples $r \in \mathbb{Z}_p$ and sets

$$X_1 := X = xG \quad \text{and} \quad X_2 := rG.$$

However, in this case the relation

$$z_0 G + z_1 X_1 + z_2 X_2 = Z$$

obtained by the output of the CDH algorithm is insufficient for recovering the DL solution x. Intuitively, the output of the CDH algorithm xrG does not give the reduction algorithm any additional information, since the reduction algorithm is able to compute $Z = xrG = rX$ by itself.

To this end, we introduce a new embedding which we call an *affine embedding*. As claimed above, when we provide a reduction for the CDH problem, the reduction algorithm gives $(\mathcal{G}, (X_1, X_2))$ to a CDH algorithm, where

$$X_1 := X = xG \quad \text{and} \quad X_2 := X + rG = (x + r)G$$

by picking a random $r \in \mathbb{Z}_p$. The affine embedding embeds the DL solution x into two group elements (X_1, X_2), where $x + r$ that is an exponent of X_2 has an affine relation of x. Then, the reduction algorithm is able to obtain non-trivial information since it is not able to compute $Z = x(x + r)G$ (or $x^2 G$) by itself. Similarly, the affine embedding is able to embed the DL solution x into multiple group elements $((x + r_1)G, \ldots, (x + r_\ell)G)$ by picking random $(r_1, \ldots, r_\ell) \in \mathbb{Z}_p^\ell$. Note that the discrete logarithm of group elements $(x, x + r_1, \ldots, x + r_\ell)$ look random in $\mathbb{Z}_p^{\ell+1}$ from Diffie-Hellman algorithm's view.

The affine embedding is still insufficient for the computational k-linear problem, i.e., given $(\mathcal{G}, X_1 := x_1 G, \ldots, X_k := x_k G, Y_1 := x_1 y_1, \ldots, Y_k := x_k y_k G)$ for random $(x_1, \ldots, x_k, y_1, \ldots, y_k)$ and compute $(y_1 + \cdots + y_k)G$. Specifically, by embedding an affine relation of the DL solution x into some $(x_1, \ldots, x_k, y_1, \ldots, y_k)$, then a relation obtained by the k-linear algorithm's output may result in a zero polynomial. To avoid the obstacle, we sample a random x_1 by ourselves and implicitly embed the DL solution x into x_1y_1. We call the embedding *implicit embedding* since we do not know a value of y_1. In other words, a k-linear algorithm enables us to obtain a non-trivial value $y_1 = x_1/x$ that enables us to provide a reduction from the DL to the k-linear problem.

1.4 Organization

In Section 2, we review several computational problems which we study in this paper. In Section 3, we review a definition of the algebraic group model (AGM) defined by Fuchsbauer et al. [FKL18]. In Sections 4 and 5, we show our technique to provide generic and tight reductions from the DL to the CDH family in cyclic groups and symmetric bilinear groups along with master theorems, respectively. In Sections 6 and 7, we show new applications of this full version, i.e., generic and tight reductions from the DL problem to the matrix computational Diffie-Hellman problem and the matrix kernel Diffie-Hellman problem, respectively.

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. Let a capital case bold letter $\mathbf{A}$ and a lower case bold letter $\mathbf{a}$ denote a matrix and a column vector, respectively. Let $\mathbf{0}_k$ denote a k-dimensional zero vector. 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 $\deg_{x,y} = 3$. As a notational convenience, we use $\deg f = 1/k$ for $f = x^{1/k}$ and $\deg f = -k$ for $f = x^{-k}$.

2 Computational Problems

In this section, we review several computational problems that we study in this paper. Specifically, in Sections 2.1, 2.2, and 2.3, we review Diffie-Hellman variants in cyclic groups, symmetric bilinear groups, and matrix Diffie-Hellman problems in asymmetric bilinear groups, respectively. The contents of this section refer to [Boy08, KSW13, MRV16, EHK$^+$17].

2.1 Diffie-Hellman Variants 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] For simplicity, when given $\mathcal{G}$ we use the notation $[a] := aG$ for $a \in \mathbb{Z}_p$.

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 $X := [x] \in \mathbb{G}; x \xleftarrow{\$} \mathbb{Z}_p$, 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 $(X_1 := [x_1], X_2 := [x_2]) \in \mathbb{G}^2; (x_1, x_2) \xleftarrow{\$} \mathbb{Z}_p^2$, compute $Z := [x_1x_2] \in \mathbb{G}$.*

[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, we study only a prime-order group in this paper as [FKL18].

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

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 $X := [x] \in \mathbb{G}; x \xleftarrow{\$} \mathbb{Z}_p$, compute $Z := [x^k] \in \mathbb{G}$.*

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

Definition 5 (k-th Root Diffie-Hellman (k-RDH) Problem)**.** *Given a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and a group element $X := [x] \in \mathbb{G}; x \xleftarrow{\$} \mathbb{Z}_p$, compute $Z := [x^{1/k}] \in \mathbb{G}$.*

The following k-Inverse Diffie-Hellman problem for $k = 1$ called the inverse computational Diffie-Hellman problem was used in [BDZ03].

Definition 6 (k-Inverse Diffie-Hellman (k-IDH) Problem)**.** *Given a group description $\mathcal{G} := (\mathbb{G}, G, p)$ and a group element $X := [x] \in \mathbb{G}; x \xleftarrow{\$} \mathbb{Z}_p$, compute $Z := [x^{-k}] \in \mathbb{G}$.*

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

Definition 7 (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*

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

compute

$$Z := [g(x_1, \ldots, x_m)] \in \mathbb{G}.$$

Note that the GDH problem contains the CDH, the k-PDH, the k-EDH, the k-RDH, and the k-IDH problem as special cases.

2.2 Diffie-Hellman Variants in Symmetric Bilinear Groups

We review computational problems in bilinear groups. For simplicity, we focus only on *symmetric* bilinear maps $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. 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 $e(xG, yG) = e(G, G)^{xy}$. For simplicity, when given $\mathcal{G}$ we use the notations $[a] := aG$ and $[a]_T := e(G, G)^a$ for $a \in \mathbb{Z}_p$.

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

Then, we summarize the CDH variants in symmetric bilinear groups.

Definition 9 (Computational Bilinear Diffie-Hellman (CBDH) Problem [BF03, Jou04])**.** *Given a bilinear group description $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ and group elements $(X_1 := [x_1], X_2 := [x_2], X_3 := [x_3]) \in \mathbb{G}^3; (x_1, x_2, x_3) \xleftarrow{\$} \mathbb{Z}_p^3$, compute $Z := [x_1x_2x_3]_T \in \mathbb{G}_T$.*

Definition 10 (k-party Bilinear Diffie-Hellman (k-PBDH) Problem)**.** *Given a group description $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ and group elements $(X_1 := [x_1], \ldots, X_k := [x_k]) \in \mathbb{G}^k; (x_1, \ldots, x_k) \xleftarrow{\$} \mathbb{Z}_p^k$, compute $Z := [x_1 \cdots x_k]_T \in \mathbb{G}_T$.*

Definition 11 (k-exponent Bilinear Diffie-Hellman (k-EBDH) Problem)**.** *Given a group description $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ and a group element $X := [x] \in \mathbb{G}; x \xleftarrow{\$} \mathbb{Z}_p$, compute $Z := [x^k]_T \in \mathbb{G}_T$.*

Definition 12 (k-th Root Bilinear Diffie-Hellman (k-RBDH) Problem)**.** *Given a group description $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ and a group element $X := [x] \in \mathbb{G}; x \xleftarrow{\$} \mathbb{Z}_p$, compute $Z := [x^{1/k}]_T \in \mathbb{G}_T$.*

Definition 13 (k-Inverse Bilinear Diffie-Hellman (k-IBDH) Problem)**.** *Given a group description $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G, e, p)$ and a group element $X := [x] \in \mathbb{G}; x \xleftarrow{\$} \mathbb{Z}_p$, compute $Z := [x^{-k}]_T \in \mathbb{G}_T$.*

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

Definition 14 (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*

$$\left(\begin{array}{l} X_1 := [f_1(x_1, \ldots, x_m, y_1, \ldots, y_n)], \ldots, X_k := [f_k(x_1, \ldots, x_m, y_1, \ldots, y_n)], \\ Y_1 := [g_1(x_1, \ldots, x_m, y_1, \ldots, y_n)]_T, \ldots, Y_\ell := [g_\ell(x_1, \ldots, x_m, y_1, \ldots, y_n)]_T \end{array} \right) \in \mathbb{G}^k \times \mathbb{G}_T^\ell;$$

$$(x_1, \ldots, x_m, y_1, \ldots, y_n) \xleftarrow{\$} \mathbb{Z}_p^{m+n},$$

compute

$$Z := [h(x_1, \ldots, x_m)]_T \in \mathbb{G}_T.$$

Note that the GBDH problem contains the CBDH, the k-PBDH, the k-EBDH, the k-RBDH, and the k-IBDH problem as special cases.

2.3 Matrix Diffie-Hellman Problem

We review matrix Diffie-Hellman problems in asymmetric bilinear groups[2]. Let $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ be an asymmetric bilinear group description, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive groups generated by G_1 and G_2 respectively, and has a prime-order p, and $\mathbb{G}_T$ is a multiplicative group of order p associated with a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, i.e., $e(G_1, G_2)$ is a generator of $\mathbb{G}_T$ and $e(xG_1, yG_2) = e(G_1, G_2)^{xy}$. For simplicity, when given $\mathcal{G}$ we use the notations $[a]_1 := aG_1$, $[a]_2 := aG_2$, and $[a]_T := e(G_1, G_2)^a$ for $a \in \mathbb{Z}_p$. Furthermore, for a matrix $\mathbf{A} = (a_{i,j})$ we use the notation $[\mathbf{A}]_1$ to denote a matrix whose every (i, j) element is $[a_{i,j}]_1$. We use analogous notations to denote $[\mathbf{A}]_2$ and $[\mathbf{A}]_T$.

Matrix Distribution. Let $\mathcal{D}_k$ be a *matrix distribution* to sample a matrix $\mathbf{A} \in \mathbb{Z}_p^{(k+1)\times k}$. Let $\bar{\mathbf{A}}$ and $\mathbf{a}^\top$ denote a top $k \times k$ submatrix and a bottom row vector of $\mathbf{A}$, respectively. Escala

[2]The problem did not appear in the preliminary version [MTT19].

et al. [EHK+17] introduced a matrix decisional Diffie-Hellman (matrix DDH) problem. Roughly speaking, the matrix DDH problem states that $([\mathbf{A}]_1, [\mathbf{As}]_1)$ and $([\mathbf{A}]_1, [\mathbf{u}]_1)$ in $\mathbb{G}_1^{(k+1)\times k} \times \mathbb{G}_1^{k+1}$ are computationally indistinguishable, where $\mathbf{A} \leftarrow \mathcal{D}_k$, $\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_p^k$, and $\mathbf{u} \overset{\$}{\leftarrow} \mathbb{Z}_p^{k+1}$. The matrix DDH contains several decisional problems as special cases. For example, we show examples of matrix distributions $\mathcal{D}_k$ for the *Symmetric k-Cascade assumption* ($\mathcal{SC}_k$), the *k-Cascade assumption* ($\mathcal{C}_k$), the *decisional k-linear assumption* ($\mathcal{L}_k$), the *incremental k-linear assumption* ($\mathcal{IL}_k$) that were introduced in [EHK+17], and the *randomized k-linear assumption* ($\mathcal{RL}_k$) that were introduced in [JR14] (they called the assumption *k-lifted assumption* and the randomized k-linear assumption was named in [MRV16]) as follows:

$$\mathcal{L}_k : \mathbf{A} = \begin{pmatrix} a_1 & & 0 \\ & \ddots & \\ 0 & & a_k \\ 1 & \cdots & 1 \end{pmatrix}, \quad \mathcal{SC}_k : \mathbf{A} = \begin{pmatrix} a & & 0 \\ 1 & \ddots & \\ & \ddots & a \\ 0 & & 1 \end{pmatrix}, \quad \mathcal{C}_k : \mathbf{A} = \begin{pmatrix} a_1 & & 0 \\ 1 & \ddots & \\ & \ddots & a_k \\ 0 & & 1 \end{pmatrix},$$

$$\mathcal{IL}_k : \mathbf{A} = \begin{pmatrix} a & & & 0 \\ & a+1 & & \\ & & \ddots & \\ 0 & & & a+k-1 \\ 1 & \cdots & & 1 \end{pmatrix}, \quad \mathcal{RL}_k : \mathbf{A} = \begin{pmatrix} a_1 & & 0 \\ & \ddots & \\ 0 & & a_k \\ a_{k+1} & \cdots & a_{2k} \end{pmatrix}.$$

We define a generalized matrix distribution to provide our master theorems as follows:

Definition 15 (Generalized Matrix Distribution $\mathcal{GM}_k$)**.** *Let $f_{i,j}(x_1, \ldots, x_m)$ be known fixed polynomials (which may include zero polynomials) for $(i, j) \in \{1, 2, \ldots, k+1\} \times \{1, 2, \ldots, k\}$. The generalized matrix distribution $\mathcal{GM}_k$ is defined by $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}_p^{(k+1)\times k}$ for*

$$a_{i,j} = f_{i,j}(x_1, \ldots, x_m); (x_1, \ldots, x_m) \overset{\$}{\leftarrow} \mathbb{Z}_p^m,$$

where

- *$\bar{\mathbf{A}}$ is full rank with overwhelming probability.*
- *There is at least one index $j \in \{1, 2, \ldots, k\}$ such that $f_{k+1,j}(x_1, \ldots, x_m)$ are non-zero polynomials.*

We use the notation $X_{i,j}$ to denote $[a_{i,j}]_1$.

We will provide generic and tight reductions for computational counter parts of matrix Diffie-Hellaman problems based on the following bilinear discrete logarithm problem in a group $\mathbb{G}_1$.

Definition 16 (Bilinear Discrete Logarithm (BDL) Problem in $\mathbb{G}_1$)**.** *Given a bilinear group description $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e, p)$ and a group element $X := [x]_1 \in \mathbb{G}_1; x \overset{\$}{\leftarrow} \mathbb{Z}_p$, compute $x \in \mathbb{Z}_p$.*

In this paper, we study computational counterparts of the matrix DDH assumption in the following two ways as [MRV16].

Matrix Computational Diffie-Hellman Problem. The first computational counterpart is the *matrix computational Diffie-Hellman (MCDH) problem.* Roughly speaking, the MCDH problem states that given $([\mathbf{A}]_1, [\bar{\mathbf{A}}\mathbf{s}]_1) \in \mathbb{G}_1^{(k+1)\times k} \times \mathbb{G}_1^{k+1}$ then computing $[\mathbf{a}^\top \mathbf{s}]_1 \in \mathbb{G}_1$ is computationally hard, where $\mathbf{A} \leftarrow \mathcal{D}_k$ and $\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_p^k$.

Definition 17 (Computational k-Linear (k-Lin) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and group elements* $(X_1 := [x_1]_1, \dots, X_k := [x_k]_1, Y_1 := [x_1 y_1]_1, \dots, Y_k := [x_k y_k]_1) \in \mathbb{G}_1^{2k+1}; (x_1, \dots, x_k, y_1, \dots, y_k) \xleftarrow{\$} \mathbb{Z}_p^{2k}$*, compute* $Z := [y_1 + \cdots + y_k]_1 \in \mathbb{G}_1$*.*

Definition 18 (Computational Symmetric k-Cascade (k-SCasc) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and group elements* $(X := [x]_1, Y_1 := [xy_1]_1, Y_2 := [y_1 + xy_2]_1, \dots, Y_k := [y_{k-1} + xy_k]_1) \in \mathbb{G}_1^{k+1}; (x, y_1, \dots, y_k) \xleftarrow{\$} \mathbb{Z}_p^{k+1}$*, compute* $Z := [y_k]_1 \in \mathbb{G}_1$*.*

Definition 19 (Computational k-Cascade (k-Casc) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and group elements* $(X_1 := [x_1]_1, \dots, X_k := [x_k]_1, Y_1 := [x_1 y_1]_1, Y_2 := [y_1 + x_2 y_2]_1, \dots, Y_k := [y_{k-1} + x_k y_k]_1) \in \mathbb{G}_1^{2k}; (x_1, \dots, x_k, y_1, \dots, y_k) \xleftarrow{\$} \mathbb{Z}_p^{2k}$*, compute* $Z := [y_k]_1 \in \mathbb{G}_1$*.*

Definition 20 (Computational Incremental k-Linear (k-IL) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and group elements* $(X := [x]_1, Y_1 := [x_1 y_1]_1, Y_2 := [(x+1)y_2]_1, \dots, Y_k := [(x+k-1)y_k]_1) \in \mathbb{G}_1^{k+1}; (x, y_1, \dots, y_k) \xleftarrow{\$} \mathbb{Z}_p^{k+1}$*, compute* $Z := [y_1 + \cdots + y_k]_1 \in \mathbb{G}_1$*.*

Definition 21 (Computational Radnomized k-Linear (k-RL) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and group elements* $(X_1 := [x_1]_1, \dots, X_{2k} := [x_{2k}]_1, Y_1 := [x_1 y_1]_1, \dots, Y_k := [x_k y_k]_1) \in \mathbb{G}_1^{3k}; (x_1, \dots, x_{2k}, y_1, \dots, y_k) \xleftarrow{\$} \mathbb{Z}_p^{3k}$*, compute* $Z := [x_{k+1} y_1 + \cdots + x_{2k} y_k]_1 \in \mathbb{G}_1$*.*

When the matrix distribution follows a generalized matrix distribution $\mathcal{GM}_k$ in Definition 15, we call the computational problem a *generalized matrix computational Diffie-Hellman* problem. We formally define it as follows.

Definition 22 (Generalized Matrix Computational Diffie-Hellman (GMCDH) Problem)**.** *Let* $f_{i,j}(x_1, \dots, x_m)$ *for* $(i,j) \in \{1, 2, \dots, k+1\} \times \{1, 2, \dots, k\}$ *be polynomials in Definition 15. Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and group elements*

$$((X_{i,j} := [f_{i,j}(x_1, \dots, x_m)]_1)_{(i,j) \in \{1,2,\dots,k+1\} \times \{1,2,\dots,k\}}, (Y_i := [\sum_{j=1}^{k} f_{i,j}(x_1, \dots, x_m) y_j]_1)_{j \in \{1,2,\dots,k\}}) \in \mathbb{G}_1^{k(k+2)};$$

$$(x_1, \dots, x_m, y_1, \dots, y_k) \xleftarrow{\$} \mathbb{Z}_p^{m+k},$$

compute

$$Z := [\sum_{j=1}^{k} f_{k+1,j}(x_1, \dots, x_m) y_j]_1 \in \mathbb{G}_1.$$

Note that the GMCDH problem contains computational variants of the k-SCasc, the k-Casc, the k-Lin, k-IL, and the k-RL problems as special cases.

Matrix Kernel Diffie-Hellman Problem. The other computational counterpart is the *matrix kernel Diffie-Hellman (MKDH) problem.* Roughly speaking, the MKDH problem states that given $[\mathbf{A}]_1 \in \mathbb{G}_1^{(k+1) \times k}$ then computing $[\mathbf{v}]_2 \in \mathbb{Z}_p^{k+1} \setminus \{\mathbf{0}_{k+1}\}$ such that $\mathbf{A}^\top \mathbf{v} = \mathbf{0}_k$ is computationally hard, where $\mathbf{A} \leftarrow \mathcal{D}_k$.

Definition 23 (Kernel k-Linear (k-Lin) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and group elements* $(X_1 := [x_1]_1, \dots, X_k := [x_k]_1) \in \mathbb{G}^k; (x_1, \dots, x_k) \xleftarrow{\$} \mathbb{Z}_p^k$*, compute* $(Z_1 := [z_1]_2, \dots, Z_{k+1} := [z_{k+1}]_2) \in \mathbb{G}_2^{k+1}$ *such that* $x_1 z_1 + z_{k+1} = \cdots = x_k z_k + z_{k+1} = 0$ *and* $(z_1, \dots, z_{k+1}) \neq \mathbf{0}_{k+1}$*.*

Definition 24 (Kernel Symmetric k-Cascade (k-SCasc) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and a group element* $X := [x]_1 \in \mathbb{G}_1; x \xleftarrow{\$} \mathbb{Z}_p$*, compute* $(Z_1 := [z_1]_2, \ldots, Z_{k+1} := [z_{k+1}]_2) \in \mathbb{G}_2^{k+1}$ *such that* $xz_1 + z_2 = \cdots = xz_k + z_{k+1} = 0$ *and* $(z_1, \ldots, z_{k+1}) \neq \mathbf{0}_{k+1}$*.*

Definition 25 (Kernel k-Cascade (k-Casc) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and group elements* $(X_1 := [x_1]_1, \ldots, X_k := [x_k]_1) \in \mathbb{G}^k; (x_1, \ldots, x_k) \xleftarrow{\$} \mathbb{Z}_p^k$*, compute* $(Z_1 := [z_1]_2, \ldots, Z_{k+1} := [z_{k+1}]_2) \in \mathbb{G}_2^{k+1}$ *such that* $x_1z_1 + z_2 = \cdots = x_kz_k + z_{k+1} = 0$ *and* $(z_1, \ldots, z_{k+1}) \neq \mathbf{0}_{k+1}$*.*

Definition 26 (Kernel Incremental k-Linear (k-IL) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and a group element* $X := [x]_1 \in \mathbb{G}_1; x \xleftarrow{\$} \mathbb{Z}_p$*, compute* $(Z_1 := [z_1]_2, \ldots, Z_{k+1} := [z_{k+1}]_2) \in \mathbb{G}_2^{k+1}$ *such that* $xz_1 + z_{k+1} = (x+1)z_2 + z_{k+1} = \cdots = (x+k-1)z_k + z_{k+1} = 0$ *and* $(z_1, \ldots, z_{k+1}) \neq \mathbf{0}_{k+1}$*.*

Definition 27 (Kernel Radnomized k-Linear (k-RL) Problem)**.** *Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and group elements* $(X_1 := [x_1]_1, \ldots, X_{2k} := [x_{2k}]_1) \in \mathbb{G}_1^{2k}; (x_1, \ldots, x_{2k}) \xleftarrow{\$} \mathbb{Z}_p^{2k}$*, compute* $(Z_1 := [z_1]_2, \ldots, Z_{k+1} := [z_{k+1}]_2) \in \mathbb{G}_2^{k+1}$ *such that* $x_1z_1 + x_{k+1}z_{k+1} = \cdots = x_kz_k + x_{2k}z_{k+1} = 0$ *and* $(z_1, \ldots, z_{k+1}) \neq \mathbf{0}_{k+1}$*.*

When the matrix distribution follows a generalized matrix distribution $\mathcal{GM}_k$ in Definition 15, we call the kernel problem a *generalized matrix kernel Diffie-Hellman* problem.

Definition 28 (Generalized Matrix Kernel Diffie-Hellman (GMKDH) Problem)**.** *Let* $f_{i,j}(x_1, \ldots, x_m)$ *for* $(i,j) \in \{1, 2, \ldots, k+1\} \times \{1, 2, \ldots, k\}$ *be polynomials in Definition 15. Given a group description* $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ *and group elements*

$$(X_{i,j} := [f_{i,j}(x_1, \ldots, x_m)]_1)_{(i,j)\in\{1,2,\ldots,k+1\}\times\{1,2,\ldots,k\}} \in \mathbb{G}_1^{k(k+1)}; (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m,$$

compute $(Z_1 := [z_1]_2, \ldots, Z_{k+1} := [z_{k+1}]_2) \in \mathbb{G}_2^{k+1}$ *such that*

$$f_{1,j}(x_1, \ldots, x_m)z_1 + \cdots + f_{k+1,j}(x_1, \ldots, x_m)z_{k+1} = 0$$

for all $j \in \{1, 2, \ldots, k\}$ *and* $(z_1, \ldots, z_{k+1}) \neq \mathbf{0}_{k+1}$*.*

Note that the GMKDH problem contains kernel variants of the k-SCasc, the k-Casc, the k-Lin, k-IL, and the k-RL problems as special cases.

3 Algebraic Group Model

In this section, we review basic notions of security games, the generic group model, and the algebraic group model. The contents of this section heavily refer to [FKL18].

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 $Z = [x_1x_2]$. 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 $\mathsf{A}_{\mathsf{gen}}$ is not given actual representations of group elements but the elements via abstract handles. For example, an adversary $\mathsf{A}_{\mathsf{gen}}$ 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 $\mathsf{A}_{\mathsf{gen}}$ is able to perform group operations only via oracle queries, e.g., a generic adversary $\mathsf{A}_{\mathsf{gen}}$ 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 output group elements are calculated with respect to given elements.

Definition 29 (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.*

4 Reductions for Diffie-Hellman Variants in Cyclic Groups

In this section, we show generic and tight reductions from the discrete logarithm (DL) problem to the computational Diffie-Hellman (CDH) problem and its variants in the algebraic group model. We first provide a *direct* reduction to the CDH in Section 4.1. Then, we provide our master theorem in Section 4.2.

4.1 DL to CDH Reduction via Affine Embedding

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 via the affine embedding.

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., $X := [x] \in \mathbb{G}$ 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 $r \overset{\$}{\leftarrow} \mathbb{Z}_p$ and compute

$$X_2 = X + rG = [x + r] \in \mathbb{G},$$

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 $[Z]_{\vec{z}}$ output 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., $Z = [x_1 x_2]$. 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

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

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

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

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	$(x_i)_{i\in\{1,2,\ldots,k\}}$	$x_1\cdots x_k$	1	k	$k \geq 2$
k-EDH	x	x^k	1	k	$k \geq 2$
k-RDH	x	$x^{1/k}$	1	$1/k$	$k \geq 2$
k-IDH	x	x^{-k}	1	$-k$	$k \geq 1$

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

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.*

4.2 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* $i \in \{1, 2, \ldots, \ell\}$*,*

(2) $\deg g(x_1, \ldots, x_m) \notin \{0, 1\}$.

Before providing a proof, we summarize the CDH variants which we summarized in Section 2.1 and the conditions of Theorem 3 in Table 1. As the table shows, CDH, k-PDH, k-EDH, k-RDH, and k-IDH simultaneously satisfy the conditions (1) and (2) in Theorem 3 (although there are restrictions of k). 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, k-RDH, and k-IDH.

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 $\mathsf{R_{gen}}$. Specifically, the reduction algorithm $\mathsf{R_{gen}}$ uses an algebraic adversary $\mathsf{A_{alg}}$ on the $\mathbf{GDH}_{\mathcal{G}}$ only once and constructs an algebraic adversary $\mathsf{B_{alg}} := \mathsf{R_{gen}^{A_{alg}}}$ 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., $X := [x] \in \mathbb{G}$ for an unknown $x \in \mathbb{Z}_p$. Thanks to the condition (1), we use the notation

$$f_i(x_1, \ldots, x_m, y_1, \ldots, y_n) = f_i^{\mathsf{L}}(y_1, \ldots, y_n) x_{\hat{i}} + f_i^{\mathsf{R}}(y_1, \ldots, y_n)$$

with some $\hat{i} \in \{1, 2, \ldots, m\}$ for all $i \in \{1, 2, \ldots, \ell\}$. 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

$$\begin{aligned} X_i &= f_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot (X + r_{\hat{i}} G) + f_i^{\mathsf{R}}(s_1, \ldots, s_n) \cdot G \\ &= [f_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot x + f_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot r_{\hat{i}} + f_i^{\mathsf{R}}(s_1, \ldots, s_n)] \in \mathbb{G} \end{aligned}$$

for all $i \in \{1, 2, \ldots, \ell\}$ by implicitly setting

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

Here, we use the notation $r_1 = 0$ for simplicity. 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 $[Z]_{\vec{z}}$ output 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., $Z = [g(x_1, \ldots, x_m)]$. 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

$$\begin{aligned} &[g(x_1, \ldots, x_m)] \\ &= z_0 G + z_1 X_1 + \cdots + z_\ell X_\ell \\ &= [z_0 + \sum_{i=1}^{\ell} z_i f_i(x_1, \ldots, x_m, y_1, \ldots, y_n)] \\ &= [z_0 + \sum_{i=1}^{\ell} z_i \left(f_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot x + f_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot r_{\hat{i}} + f_i^{\mathsf{R}}(s_1, \ldots, s_n) \right)] \\ &= [\left(\sum_{i=1}^{\ell} z_i f_i^{\mathsf{L}}(s_1, \ldots, s_n) \right) x + z_0 + \sum_{i=1}^{\ell} z_i \left(f_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot r_{\hat{i}} + f_i^{\mathsf{R}}(s_1, \ldots, s_n) \right)]. \end{aligned}$$

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

$$\begin{aligned} &g(x, x + r_2, \ldots, x + r_m) \\ = &\left(\sum_{i=1}^{\ell} z_i f_i^{\mathsf{L}}(s_1, \ldots, s_n) \right) x + z_0 + \sum_{i=1}^{\ell} z_i \left(f_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot r_{\hat{i}} + f_i^{\mathsf{R}}(s_1, \ldots, s_n) \right) \mod p. \end{aligned}$$

Observe that a degree of the left hand side with respect to a variable x is not in $\{0, 1\}$ thanks to the condition (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 $[x] = X$, 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 7 is $(t^2/p, t)$-hard in the generic group model if the conditions (1) and (2) of Theorem 3 simultaneously hold.*

5 Reductions for Diffie-Hellman Variants in Symmetric Bilinear Groups

In this section, we show generic and tight reductions from the bilinear discrete logarithm (BDL) problem to the computational bilinear Diffie-Hellman (CBDH) problem and its variants in an *algebraic bilinear group model* which we define in Section 5.1. In Section 5.2, we provide a reduction from the BDL to the CBDH. Finally, we provide our master theorem in Section 5.3.

5.1 Algebraic Symmetric Bilinear Group Model

In advance of the reduction, we define an algebraic symmetric bilinear algorithm. The definition is analogous to Definition 29 in the sense that the algebraic symmetric 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 symmetric bilinear algorithm should also output a *representation* which indicates how output 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 30 (Algebraic Symmetric 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 $\vec{z} := ((z_{i,j})_{0\le i\le j\le k}, (z'_i)_{1\le i\le \ell}) \in \mathbb{Z}_p^{\frac{(k+1)(k+2)}{2}+\ell}$ such that $Z = \left(\prod_{0\le i\le j\le k} e(X_i, X_j)^{z_{i,j}}\right) \cdot \left(\prod_{i=1}^{\ell} Y_i^{z'_i}\right)$. We denote such an output as $[Z]_{\vec{z}}$.*

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

5.2 BDL to CBDH Reduction via Affine Embedding

In this subsection, we extend the approach in Section 4 and prove the following reduction via the affine embedding.

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., $X := [x] \in \mathbb{G}$ 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

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

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 CBDH 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 $[Z]_{\vec{z}}$ output 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 CBDH, i.e., $Z = e(G, G)^{x_1 x_2 x_3}$. It holds with probability $\mathsf{Adv}^{\mathbf{BDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$. Then, we use $X_0 := [1]$ for notational convenience and the representation vector $\vec{z} := (z_{i,j})_{0 \le i \le j \le 3}$ satisfies

$$\begin{aligned}
[x_1 x_2 x_3]_T &= [x(x+r)(x+s)]_T \\
&= \prod_{0 \le i \le j \le 3} e(X_i, X_j)^{z_{i,j}} \\
&= [z_{0,0} + z_{0,1}x + z_{0,2}(x+r) + z_{0,3}(x+s) + z_{1,1}x^2 + z_{1,2}x(x+r) + z_{1,3}x(x+s) \\
&\quad + z_{2,2}(x+r)^2 + z_{2,3}(x+r)(x+s) + z_{3,3}(x+s)^2]_T.
\end{aligned}$$

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

$$\begin{aligned}
& x(x+r)(x+s) \\
= \; & z_{0,0} + z_{0,1}x + z_{0,2}(x+r) + z_{0,3}(x+s) + z_{1,1}x^2 + z_{1,2}x(x+r) \\
& + z_{1,3}x(x+s) + z_{2,2}(x+r)^2 + z_{2,3}(x+r)(x+s) + z_{3,3}(x+s)^2 \mod p \\
\Leftrightarrow \; & x^3 + (r + s - z_{1,1} - z_{1,2} - z_{1,3} - z_{2,2} - z_{2,3} - z_{3,3})x^2 \\
& + (rs - z_{0,1} - z_{0,2} - z_{0,3} - rz_{1,2} - sz_{1,3} - 2rz_{2,2} - (r+s)z_{2,3} - 2sz_{3,3})x \\
& - z_{0,0} - rz_{0,2} - sz_{0,3} - r^2 z_{2,2} - rsz_{2,3} - s^2 z_{3,3} = 0 \mod p.
\end{aligned}$$

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 $[x] = X$, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ successfully finds a correct solution of the $\mathbf{BDL}_{\mathcal{G}}$. $\square$

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

Theorem 6 (Generic Hardness of CBDH)**.** *The computational bilinear Diffie-Hellman problem in Definition 9 is $(t^2/p, t)$-hard in the generic group model.*

Table 2: Applicability of our technique in symmetric bilinear groups

problem	$(f_i)_{i\in\{1,2,\dots,k\}}$	h	max deg f_i	deg h	reduction?
CBDH	$(x_i)_{i\in[3]}$	$x_1x_2x_3$	1	3	Yes
k-PBDH	$(x_i)_{i\in\{1,2,\dots,k\}}$	$x_1\cdots x_k$	1	k	$k\geq 3$
k-EBDH	x	x^k	1	k	$k\geq 3$
k-RBDH	x	$x^{1/k}$	1	$1/k$	$k\geq 2$
k-IBDH	x	x^{-k}	1	$-k$	$k\geq 1$

5.3 Master Theorem in Symmetric 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) $\deg_{x_1,\dots,x_m} f_i(x_1,\dots,x_m,y_1,\dots,y_n) \in \{0,1\}$ *for all* $i\in\{1,2,\dots,k\}$,

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

(3) $\deg h(x_1,\dots,x_m) \notin \{0,1,2\}$.

Before providing a proof, we summarize the CBDH variants which we summarized in Section 2.2 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, CBDH, k-PBDH, k-EBDH, k-RBDH, and k-IBDH simultaneously satisfy the conditions (1) and (3) in Theorem 7 (although there are restrictions of k). Hence, as immediate corollary of the master theorem, we are able to provide generic and tight reductions from the BDL to the k-PBDH, k-EBDH, k-RBDH, and k-IBDH.

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.

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 constructs 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., $X := [x] \in \mathbb{G}$ for an unknown $x \in \mathbb{Z}_p$. Thanks to the condition (1), we use the notation

$$f_i(x_1,\dots,x_m,y_1,\dots,y_n) = f_i^{\mathsf{L}}(y_1,\dots,y_n)x_{\hat{i}} + f_i^{\mathsf{R}}(y_1,\dots,y_n)$$

with some $\hat{i} \in \{1,2,\dots,m\}$ for all $i \in \{1,2,\dots,k\}$. Thanks to the condition (2), we use the notation

$$g_i(x_1,\dots,x_m,y_1,\dots,y_n) = g_i^{\mathsf{L}}(y_1,\dots,y_n)x_{\hat{i}_1}x_{\hat{i}_2} + g_i^{\mathsf{M}}(y_1,\dots,y_n)x_{\hat{i}_3} + g_i^{\mathsf{R}}(y_1,\dots,y_n)$$

with some $\hat{i}_1, \hat{i}_2, \hat{i}_3 \in \{1, 2, \ldots, m\}$ for all $i \in \{1, 2, \ldots, \ell\}$. 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

$$\begin{aligned} X_i &= f_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot (X + r_{\hat{i}} G) + f_i^{\mathsf{R}}(s_1, \ldots, s_n) \cdot G \\ &= [f_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot (x + r_{\hat{i}}) + f_i^{\mathsf{R}}(s_1, \ldots, s_n)] \in \mathbb{G} \end{aligned}$$

for all $i \in \{1, 2, \ldots, k\}$ and

$$\begin{aligned} Y_i &= e(X + r_{\hat{i}_1} G, X + r_{\hat{i}_2} G)^{g_i^{\mathsf{L}}(s_1,\ldots,s_n)} \cdot e(G, X + r_{\hat{i}_3} G)^{g_i^{\mathsf{M}}(s_1,\ldots,s_n)} \cdot e(G, G)^{g_i^{\mathsf{R}}(s_1,\ldots,s_n)} \\ &= [g_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot (x + r_{\hat{i}_1})(x + r_{\hat{i}_2}) + g_i^{\mathsf{M}}(s_1, \ldots, s_n) \cdot (x + r_{\hat{i}_3}) + g_i^{\mathsf{R}}(s_1, \ldots, s_n)]_T \in \mathbb{G}_T \end{aligned}$$

for all $i \in \{1, 2, \ldots, \ell\}$ by implicitly setting

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

Here, we use the notation $r_1 = 0$ for simplicity. 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 $[Z]_{\vec{z}}$ output 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., $Z = [h(x_1, \ldots, x_m)]_T$. It holds with probability $\mathsf{Adv}^{\mathbf{GBDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$. Then, the representation vector $\vec{z} := ((z_{i,j})_{0 \le i \le j \le k}, (z'_i)_{1 \le i \le \ell})$ satisfies

$$\begin{aligned} & [h(x_1, \ldots, x_m)]_T \\ &= \left(\prod_{0 \le i \le j \le k} e(X_i, X_j)^{z_{i,j}} \right) \cdot \left(\prod_{1 \le i \le \ell} Y_i^{z'_i} \right) \\ &= [\sum_{0 \le i \le j \le k} z_{i,j} f_i(x_1, \ldots, x_m, y_1, \ldots, y_n) \cdot f_j(x_1, \ldots, x_m, y_1, \ldots, y_n) + \sum_{i=1}^{\ell} z'_i g_i(x_1, \ldots, x_m, y_1, \ldots, y_n)]_T \\ &= [\sum_{0 \le i \le j \le k} z_{i,j} \left(f_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot (x + r_{\hat{i}}) + f_i^{\mathsf{R}}(s_1, \ldots, s_n) \right) \left(f_j^{\mathsf{L}}(s_1, \ldots, s_n) \cdot (x + r_{\hat{j}}) + f_j^{\mathsf{R}}(s_1, \ldots, s_n) \right) \\ &\quad + \sum_{i=1}^{\ell} z'_i \left(g_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot (x + r_{\hat{i}_1})(x + r_{\hat{i}_2}) + g_i^{\mathsf{M}}(s_1, \ldots, s_n) \cdot (x + r_{\hat{i}_3}) + g_i^{\mathsf{R}}(s_1, \ldots, s_n) \right)]_T \\ &= [\left(\sum_{0 \le i \le j \le k} z_{i,j} f_i^{\mathsf{L}}(s_1, \ldots, s_n) f_j^{\mathsf{L}}(s_1, \ldots, s_n) + \sum_{i=1}^{\ell} z'_i g_i^{\mathsf{L}}(s_1, \ldots, s_n) \right) x^2 \\ &\quad + \left(\sum_{0 \le i \le j \le k} z_{i,j} \left(f_i^{\mathsf{L}}(s_1, \ldots, s_n)(f_j^{\mathsf{L}}(s_1, \ldots, s_n) \cdot r_{\hat{j}} + f_j^{\mathsf{R}}(s_1, \ldots, s_n)) \right. \right. \\ &\qquad\qquad \left. + f_j^{\mathsf{L}}(s_1, \ldots, s_n)(f_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot r_{\hat{i}} + f_i^{\mathsf{R}}(s_1, \ldots, s_n)) \right) \\ &\qquad \left. + \sum_{i=1}^{\ell} z'_i \left(g_i^{\mathsf{L}}(s_1, \ldots, s_n) \cdot (r_{\hat{i}_1} + r_{\hat{i}_2}) + g_i^{\mathsf{M}}(s_1, \ldots, s_n) \right) \right) x \end{aligned}$$

$$+ \sum_{0\le i\le j\le k} z_{i,j} \left(f_i^{\mathsf{L}}(s_1,\ldots,s_n)\cdot r_{\hat{i}} + f_i^{\mathsf{R}}(s_1,\ldots,s_n)\right)\left(f_j^{\mathsf{L}}(s_1,\ldots,s_n)\cdot r_{\hat{j}} + f_j^{\mathsf{R}}(s_1,\ldots,s_n)\right)$$
$$+ \sum_{i=1}^{\ell} z_i' \left(g_i^{\mathsf{L}}(s_1,\ldots,s_n)\cdot r_{\hat{i}_1} r_{\hat{i}_2} + g_i^{\mathsf{M}}(s_1,\ldots,s_n)\cdot r_{\hat{i}_3} + g_i^{\mathsf{R}}(s_1,\ldots,s_n)\right)]_T.$$

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

$$\begin{aligned}
&h(x, x+r_2,\ldots,x+r_m)\\
&= \left(\sum_{0\le i\le j\le k} z_{i,j} f_i^{\mathsf{L}}(s_1,\ldots,s_n) f_j^{\mathsf{L}}(s_1,\ldots,s_n) + \sum_{i=1}^{\ell} z_i' g_i^{\mathsf{L}}(s_1,\ldots,s_n)\right) x^2\\
&\quad + \left(\sum_{0\le i\le j\le k} z_{i,j}\left(f_i^{\mathsf{L}}(s_1,\ldots,s_n)(f_j^{\mathsf{L}}(s_1,\ldots,s_n)\cdot r_{\hat{j}} + f_j^{\mathsf{R}}(s_1,\ldots,s_n))\right.\right.\\
&\qquad\qquad \left. + f_j^{\mathsf{L}}(s_1,\ldots,s_n)(f_i^{\mathsf{L}}(s_1,\ldots,s_n)\cdot r_{\hat{i}} + f_i^{\mathsf{R}}(s_1,\ldots,s_n))\right)\\
&\qquad \left. + \sum_{i=1}^{\ell} z_i' \left(g_i^{\mathsf{L}}(s_1,\ldots,s_n)\cdot(r_{\hat{i}_1} + r_{\hat{i}_2}) + g_i^{\mathsf{M}}(s_1,\ldots,s_n)\right)\right) x\\
&\quad + \sum_{0\le i\le j\le k} z_{i,j}\left(f_i^{\mathsf{L}}(s_1,\ldots,s_n)\cdot r_{\hat{i}} + f_i^{\mathsf{R}}(s_1,\ldots,s_n)\right)\left(f_j^{\mathsf{L}}(s_1,\ldots,s_n)\cdot r_{\hat{j}} + f_j^{\mathsf{R}}(s_1,\ldots,s_n)\right)\\
&\quad + \sum_{i=1}^{\ell} z_i'\left(g_i^{\mathsf{L}}(s_1,\ldots,s_n)\cdot r_{\hat{i}_1} r_{\hat{i}_2} + g_i^{\mathsf{M}}(s_1,\ldots,s_n)\cdot r_{\hat{i}_3} + g_i^{\mathsf{R}}(s_1,\ldots,s_n)\right) \mod p.
\end{aligned}$$

Observe that a degree of the left hand side with respect to a variable x is not in $\{0,1,2\}$ thanks to the condition (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 $[x] = X$, 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 14 is $(t^2/p, t)$-hard in the generic group model if the conditions (1)–(3) of Theorem 7 simultaneously hold.*

6 Reduction for Matrix Computational Diffie-Hellman Problem

In this section, we show generic and tight reductions from the bilinear discrete logarithm (BDL) problem in $\mathbb{G}_1$ to the matrix computational Diffie-Hellman (MCDH) problem in an *algebraic bilinear group model specific to the MCDH* which we define in Section 6.1. In Section 6.2, we provide a reduction from the BDL in $\mathbb{G}_1$ to the computational k-linear problem which is a special case of the MCDH via the implicit embedding. Finally, we provide our master theorem for the MCDH in Section 6.3.

6.1 Algebraic Group Model for Matrix Computational Diffie-Hellman Problem

In advance of the reduction, we define an algebraic algorithm for the MCDH problem. We define the notion to be compatible with the MCDH problem so that both inputs and outputs of the problems are group elements in $\mathbb{G}_1$. The definition is almost the same as Definition 29 in the sense that the algebraic algorithm is able to output only group elements which are derived from group additions in $\mathbb{G}_1$ of given elements. Furthermore, the algebraic algorithm should also output a *representation* which indicates how output group elements are calculated. We formally provide a definition as follows.

Definition 31 (Algebraic Algorithm for MCDH)**.** *An algorithm $\mathsf{A}_{\mathsf{alg}}$ executed in an algebraic security game $\mathbf{G}_{\mathcal{G}}$ in an asymmetric bilinear group $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ is called* algebraic *if for all group elements $Z \in \mathbb{G}_1$ that $\mathsf{A}_{\mathsf{alg}}$ outputs, it additionally returns the representation of Z with respect to given group elements. Specifically, if $\vec{X} := (X_0, \dots, X_\ell) \in \mathbb{G}_1^{\ell+1}$, where $X_0 := G_1$, 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 $\vec{z} := (z_i)_{0 \le i \le \ell} \in \mathbb{Z}_p^{\ell+1}$ such that $Z = \sum_{i=0}^{\ell} z_i X_i$. We use $[Z]_{\vec{z}}$ to denote such an output.*

To be precise, the definition captures not only the MCDH problem but also any computational problem all of whose inputs and outputs are in $\mathbb{G}_1$.

6.2 BDL to Computational k-Lin Reduction via Implicit Embedding

Theorem 9. $\mathbf{BDL}_{\mathcal{G}} \Rightarrow_{\mathsf{alg}} k\text{-}\mathbf{Lin}_{\mathcal{G}}$.

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 k-$\mathbf{Lin}_{\mathcal{G}}$ only once and construct an algebraic adversary $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ on the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$.

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ is given a group description $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e, p)$ and an instance of the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$, i.e., $X := [x]_1 \in \mathbb{G}$ for an unknown $x \in \mathbb{Z}_p$. Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ creates an instance of the k-$\mathbf{Lin}_{\mathcal{G}}$ as follows: Pick random $(c, x_2, \dots, x_k, y_2, \dots, y_k) \xleftarrow{\$} \mathbb{Z}_p^{2k+1}$. If $c = 0$, then resample the value. Otherwise, compute

$$Y_1 := cG_1 \in \mathbb{G}_1,$$

and

$$X_i := [x_i]_1, \qquad Y_i := [x_i y_i]_1$$

for all $i \in \{2, k\}$, then set

$$(X_1 = X, X_2, \dots, X_k, Y_1, \dots, Y_k) \in \mathbb{G}_1^{2k}$$

by implicitly setting

$$(x_1, x_2, \dots, x_k, y_1, y_2, \dots, y_k) = (x, x_2, \dots, x_k, c/x, y_2, \dots, y_k).$$

Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ gives a bilinear group description $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e, p)$ and group elements $(X_1, \dots, X_k, Y_1, \dots, Y_k) \in \mathbb{G}_1^{2k}$ to $\mathsf{A}_{\mathsf{alg}}$. Observe that $(X_1, \dots, X_k, Y_1, \dots, Y_k)$ is a valid k-Lin instance 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 $\mathsf{Adv}^{k\text{-}\mathbf{Lin}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$ and a running time $\mathsf{Time}^{k\text{-}\mathbf{Lin}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$.

Next, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses $[Z]_{\vec{z}}$ output by an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the k-$\mathbf{Lin}_{\mathcal{G}}$ and computes a solution of the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$. Assume the output is a correct solution of the k-Lin, i.e., $Z = [y_1 + \cdots + y_k]_1$. It holds with probability $\mathsf{Adv}^{k\text{-}\mathbf{Lin}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$. Then, the representation vector $\vec{z} := (z_0, z_1, \ldots, z_{2k})$ satisfies

$$\begin{aligned}
[y_1 + \cdots + y_k]_1 &= [c/x + y_2 + \cdots + y_k]_1 \\
&= z_0 G_1 + z_1 X_1 + \cdots + z_k X_k + z_{k+1} Y_1 + \cdots + z_{2k} Y_k \\
&= [z_0 + z_1 x + z_2 x_2 + \cdots + z_k x_k + z_{k+1} c + z_{k+2} y_2 + \cdots + z_{2k} y_{2k}]_1
\end{aligned}$$

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

$$\begin{aligned}
&c/x + y_2 + \cdots + y_k = z_0 + z_1 x + z_2 x_2 + \cdots + z_k x_k + z_{k+1} c + z_{k+2} y_2 + \cdots + z_{2k} y_{2k} \mod p \\
\Leftrightarrow\ & z_1 x^2 + (z_0 + z_2 x_2 + \cdots + z_k x_k + z_{k+1} c + (z_{k+2} - 1) y_2 + \cdots + (z_{2k} - 1) y_{2k}) x - c = 0 \mod p.
\end{aligned}$$

Observe that the polynomial has to be a non-zero polynomial due to the non-zero constant term c. Since the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ knows values of $(c, x_2, \ldots, x_k, y_2, \ldots, y_k)$, it is able to find all solutions for x in polynomial time. By checking $[x]_1 = X$, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ successfully finds a correct solution of the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$. □

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

Theorem 10 (Generic Hardness of k-Lin)**.** *The computational k-linear problem in Definition 17 is $(t^2/p, t)$-hard in the generic group model.*

6.3 Master Theorem for Matrix Computational Diffie-Hellman Problem

In this subsection, we provide the following master theorem for MCDH problems to indicate the power of our technique.

Theorem 11 (Master Theorem for the MCDH Problem)**.** $\mathbf{DL}_{\mathcal{G}} \Rightarrow_{\mathsf{alg}} \mathbf{GMCDH}_{\mathcal{G}}$ *holds when the following conditions are simultaneously satisfied: There is at least one tuple of indices $(\hat{\imath}, \hat{\jmath}, \hat{\ell}) \in \{1, 2, \ldots, k\} \times \{1, 2, \ldots, k\} \times \{1, 2, \ldots, m\}$ such that*

(1) $\deg_{x_{\hat{\ell}}} f_{i,j}(x_1, \ldots, x_m) \in \{0, 1\}$ *for all* $(i, j) \in \{1, 2, \ldots, k+1\} \times \{1, 2, \ldots, k\}$,

(2) $\deg_{x_{\hat{\ell}}} f_{\hat{\imath},\hat{\jmath}}(x_1, \ldots, x_m) = 1$,

(3) All monomials of non-zero polynomial $f_{i,\hat{\jmath}}(x_1, \ldots, x_m)$ have a variable $x_{\hat{\ell}}$ for all $i \in \{1, 2, \ldots, k\}$,

(4) $f_{k+1,\hat{\jmath}}(x_1, \ldots, x_m)$ has a non-zero monomial that does not have $x_{\hat{\ell}}$.

Before providing a proof, we summarize the MCDH variants which we summarized in Section 2.3 and the conditions of Theorem 11 in Table 3. As the table shows, k-Lin, k-SCasc, k-Casc, k-IL, and k-RL simultaneously satisfy the conditions (1)–(4) in Theorem 11. Hence, as immediate corollary of the master theorem, we are able to provide generic and tight reductions from the BDL to the k-SCasc, k-Casc, k-IL, and k-RL.

Then, we show a proof of Theorem 11. In advance, we claim that the conditions (1)–(3) will be used to ensure that the reduction algorithm is able to produce all group elements of the GMCDH during a reduction, while all the conditions (1)–(4) will be used to ensure that the modular equation never becomes a zero polynomial.

Table 3: Applicability of our technique in cyclic groups

problem	$\hat{i}$	$\hat{j}$	$x_{\hat{\ell}}$	reduction?
k-Lin	$\{1,2,\dots,k\}$	$\hat{i}$	$a_{\hat{i}}$	Yes
k-SCasc	$\{1,2,\dots,k\}$	$\hat{i}$	a	Yes
k-Casc	$\{1,2,\dots,k\}$	$\hat{i}$	$a_{\hat{i}}$	Yes
k-IL	$\{1,2,\dots,k\}$	$\hat{i}$	$a+\hat{i}-1$	Yes
k-RL	$\{1,2,\dots,k\}$	$\hat{i}$	$a_{\hat{i}}$	Yes

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 $\mathbf{GMCDH}_{\mathcal{G}}$ only once and constructs an algebraic adversary $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ on the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$.

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ is given a bilinear group description $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e, p)$ and an instance of the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$, i.e., $X := [x]_1 \in \mathbb{G}$ for an unknown $x \in \mathbb{Z}_p$. Thanks to the condition (1), we use the notation

$$f_{i,j}(x_1,\dots,x_m) = f^{\mathsf{L}}_{i,j}(x_1,\dots,x_{\hat{\ell}-1},x_{\hat{\ell}+1},\dots,x_m)x_{\hat{\ell}} + f^{\mathsf{R}}_{i,j}(x_1,\dots,x_{\hat{\ell}-1},x_{\hat{\ell}+1},\dots,x_m)$$

for all $(i,j) \in \{1,2,\dots,k+1\} \times \{1,2,\dots,k\}$. Thanks to the condition (2),

$$f^{\mathsf{L}}_{\hat{i},\hat{j}}(x_1,\dots,x_{\hat{\ell}-1},x_{\hat{\ell}+1},\dots,x_m) \neq 0$$

holds. Thanks to the condition (3),

$$f^{\mathsf{R}}_{i,\hat{j}}(x_1,\dots,x_{\hat{\ell}-1},x_{\hat{\ell}+1},\dots,x_m) = 0$$

holds for all $i \in \{1,2,\dots,k\}$. Thanks to the condition (4),

$$f^{\mathsf{R}}_{k+1,\hat{j}}(x_1,\dots,x_{\hat{\ell}-1},x_{\hat{\ell}+1},\dots,x_m) \neq 0$$

holds. Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ creates an instance of the $\mathbf{GMCDH}_{\mathcal{G}}$ as follows: Pick random $(c, r_1,\dots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\cdots,r_m,s_1,\dots,s_{\hat{j}-1},s_{\hat{j}+1},\cdots,s_k) \xleftarrow{\$} \mathbb{Z}_p^{m+k-1}$. If $c = 0$ or $f^{\mathsf{R}}_{k+1,\hat{j}}(r_1,\dots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\dots,r_m) = 0$, then resample the values. Otherwise, compute

$$\begin{aligned} X_i &= f^{\mathsf{L}}_{i,j}(r_1,\dots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\dots,r_m)\cdot X + f^{\mathsf{R}}_{i,j}(r_1,\dots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\dots,r_m)\cdot G_1 \\ &= [f^{\mathsf{L}}_{i,j}(r_1,\dots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\dots,r_m)\cdot x + f^{\mathsf{R}}_{i,j}(r_1,\dots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\dots,r_m)]_1 \in \mathbb{G}_1 \end{aligned}$$

for all $(i,j) \in \{1,2,\dots,k+1\} \times \{1,2,\dots,k\}$,

$$\begin{aligned} Y_j = &\sum_{j=1}^{\hat{j}-1}\left(f^{\mathsf{L}}_{i,j}(r_1,\dots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\dots,r_m)\cdot X + f^{\mathsf{R}}_{i,j}(r_1,\dots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\dots,r_m)\cdot G_1\right)s_j \\ &+ f^{\mathsf{L}}_{i,\hat{j}}(r_1,\dots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\dots,r_m)\cdot cG_1 \\ &+ \sum_{j=\hat{j}+1}^{k}\left(f^{\mathsf{L}}_{i,j}(r_1,\dots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\dots,r_m)\cdot X + f^{\mathsf{R}}_{i,j}(r_1,\dots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\dots,r_m)\cdot G_1\right)s_j \end{aligned}$$

$$= [\sum_{j=1}^{k} \left(f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \cdot x + f^{\mathsf{R}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \right) s_j]_1 \in \mathbb{G}_1$$

for all $i \in \{1, 2, \ldots, k\}$ by implicitly setting

$$\begin{aligned}&(x_1, \ldots, x_{\hat{\ell}-1}, x_{\hat{\ell}}, x_{\hat{\ell}+1}, \cdots, x_m, y_1, \ldots, y_{\hat{j}-1}, y_{\hat{j}}, y_{\hat{j}+1}, \cdots, y_k)\\ &= (r_1, \ldots, r_{\hat{\ell}-1}, x, r_{\hat{\ell}+1}, \cdots, r_m, s_1, \ldots, s_{\hat{j}-1}, c/x, s_{\hat{j}+1}, \cdots, s_k).\end{aligned}$$

Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ gives a bilinear group description $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e, p)$ and group elements $((X_{i,j})_{(i,j) \in \{1,2,\ldots,k+1\} \times \{1,2,\ldots,k\}}, (Y_i)_{i \in \{1,2,\ldots,k\}}) \in \mathbb{G}_1^{k(k+2)}$ to $\mathsf{A}_{\mathsf{alg}}$. Observe that a set of group elements is a valid GMCDH instance since $y_{\hat{j}}$ is independently distributed of x 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 $\mathsf{Adv}^{\mathbf{GMCDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$ and a running time $\mathsf{Time}^{\mathbf{GMCDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$.

Next, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses $[Z]_{\vec{z}}$ output by an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $\mathbf{GMCDH}_{\mathcal{G}}$ and computes a solution of the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$. Assume the output is a correct solution of the GMCDH, i.e., $Z = [\sum_{j=1}^{k} f_{k+1,j}(x_1, \ldots, x_m) y_j]_1$. It holds with probability $\mathsf{Adv}^{\mathbf{GMCDH}_{\mathcal{G}}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$. Then, the representation vector $\vec{z} := (z_0, (z_{i,j})_{(i,j) \in \{1,2,\ldots,k+1\} \times \{1,2,\ldots,k\}}, (z'_i)_{i \in \{1,2,\ldots,k\}})$ satisfies

$$\begin{aligned}
&[\sum_{j=1}^{k} f_{k+1,j}(x_1, \ldots, x_m) y_j]_1 \\
&= z_0 G_1 + \sum_{j=1}^{k} \sum_{i=1}^{k+1} z_{i,j} X_{i,j} + \sum_{i=1}^{k} z'_i Y_i \\
&= [z_0 + \sum_{j=1}^{k} \sum_{i=1}^{k+1} z_{i,j} \left(f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) x + f^{\mathsf{R}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \right) \\
&\quad + \sum_{i=1}^{k} z'_i \cdot \sum_{j=1}^{\hat{j}-1} \left(f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) x + f^{\mathsf{R}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \right) s_j \\
&\quad + \sum_{i=1}^{k} z'_i f^{\mathsf{L}}_{i,\hat{j}}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) c \\
&\quad + \sum_{i=1}^{k} z'_i \cdot \sum_{j=\hat{j}+1}^{k} \left(f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) x + f^{\mathsf{R}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \right) s_j]_1 \\
&= [\Bigg(\sum_{j=1}^{k} \sum_{i=1}^{k+1} z_{i,j} f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) + \sum_{j=1}^{k} \sum_{i=1}^{\hat{i}-1} z'_i f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) s_j \\
&\quad + \sum_{j=1}^{k} \sum_{i=\hat{i}+1}^{k} z'_i f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) s_j \Bigg) x \\
&\quad + z_0 + \sum_{j=1}^{k} \sum_{i=1}^{k+1} z_{i,j} f^{\mathsf{R}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) + \sum_{i=1}^{k} \sum_{j=1}^{\hat{j}-1} z'_i f^{\mathsf{R}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) s_j \\
&\quad + \sum_{i=1}^{k} z'_i f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) c + \sum_{i=1}^{k} \sum_{j=\hat{j}+1}^{k} z'_i f^{\mathsf{R}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) s_j]_1.
\end{aligned}$$

Furthermore,

$$
\begin{aligned}
&\sum_{j=1}^{k} f_{k+1,j}(x_1,\ldots,x_m)y_j \\
&= \sum_{j=1}^{\hat{j}-1}\left(f^{\mathsf{L}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)x + f^{\mathsf{R}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)\right)s_j \\
&\quad + f^{\mathsf{R}}_{k+1,\hat{j}}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)c + f^{\mathsf{R}}_{k+1,\hat{j}}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)(c/x) \\
&\quad + \sum_{j=\hat{j}+1}^{k}\left(f^{\mathsf{L}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)x + f^{\mathsf{R}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)\right)s_j \\
&= \left(\sum_{j=1}^{\hat{j}-1} f^{\mathsf{L}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j + \sum_{j=\hat{j}+1}^{k} f^{\mathsf{L}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j\right)x \\
&\quad + \sum_{j=1}^{\hat{j}-1} f^{\mathsf{R}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j + f^{\mathsf{R}}_{k+1,\hat{j}}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)c \\
&\quad + \sum_{j=\hat{j}+1}^{k} f^{\mathsf{R}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j \\
&\quad + f^{\mathsf{R}}_{k+1,\hat{j}}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)(c/x)
\end{aligned}
$$

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

$$
\begin{aligned}
&\left(\sum_{j=1}^{\hat{j}-1} f^{\mathsf{L}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j + \sum_{j=\hat{j}+1}^{k} f^{\mathsf{L}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j\right)x \\
&+ \sum_{j=1}^{\hat{j}-1} f^{\mathsf{R}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j + f^{\mathsf{R}}_{k+1,\hat{j}}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)c \\
&+ \sum_{j=\hat{j}+1}^{k} f^{\mathsf{R}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j \\
&+ f^{\mathsf{R}}_{k+1,\hat{j}}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)(c/x) \\
= &\left(\sum_{j=1}^{k}\sum_{i=1}^{k+1} z_{i,j} f^{\mathsf{L}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m) + \sum_{j=1}^{k}\sum_{i=1}^{\hat{i}-1} z'_i f^{\mathsf{L}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j\right. \\
&\left.+\sum_{j=1}^{k}\sum_{i=\hat{i}+1}^{k} z'_i f^{\mathsf{L}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j\right)x \\
&+ z_0 + \sum_{j=1}^{k}\sum_{i=1}^{k+1} z_{i,j} f^{\mathsf{R}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m) + \sum_{i=1}^{k}\sum_{j=1}^{\hat{j}-1} z'_i f^{\mathsf{R}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j
\end{aligned}
$$

$$
\begin{aligned}
&+\sum_{i=1}^{k} z'_i f^{\mathsf{L}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)c+\sum_{i=1}^{k}\sum_{j=\hat{j}+1}^{k} z'_i f^{\mathsf{R}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j \mod p\\
\Leftrightarrow &\left(\sum_{j=1}^{\hat{j}-1} f^{\mathsf{L}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j+\sum_{j=\hat{j}+1}^{k} f^{\mathsf{L}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j\right.\\
&-\sum_{j=1}^{k}\sum_{i=1}^{k+1} z_{i,j} f^{\mathsf{L}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)-\sum_{j=1}^{k}\sum_{i=1}^{\hat{i}-1} z'_i f^{\mathsf{L}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j\\
&\left.-\sum_{j=1}^{k}\sum_{i=\hat{i}+1}^{k} z'_i f^{\mathsf{L}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j\right)x^2\\
&+\left(\sum_{j=1}^{\hat{j}-1} f^{\mathsf{R}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j+f^{\mathsf{R}}_{k+1,\hat{j}}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)c\right.\\
&+\sum_{j=\hat{j}+1}^{k} f^{\mathsf{R}}_{k+1,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j-z_0\\
&+\sum_{j=1}^{k}\sum_{i=1}^{k+1} z_{i,j} f^{\mathsf{R}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)+\sum_{i=1}^{k}\sum_{j=1}^{\hat{j}-1} z'_i f^{\mathsf{R}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j\\
&\left.+\sum_{i=1}^{k} z'_i f^{\mathsf{L}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)c+\sum_{i=1}^{k}\sum_{j=\hat{j}+1}^{k} z'_i f^{\mathsf{R}}_{i,j}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)s_j\right)x\\
&+cf^{\mathsf{R}}_{k+1,\hat{j}}(r_1,\ldots,r_{\hat{\ell}-1},r_{\hat{\ell}+1},\ldots,r_m)=0 \mod p.
\end{aligned}
$$

Observe that the polynomial has to be a non-zero polynomial due to the non-zero constant term c and non-zero $h_{i,\hat{j}}(x_1,\ldots,x_{\hat{\ell}-1},x_{\hat{\ell}+1},\ldots,x_m)$. Since the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ knows values of $(c,x_1,\ldots,x_{\hat{\ell}-1},x_{\hat{\ell}+1},\cdots,x_m,y_1,\ldots,y_{\hat{j}-1},y_{\hat{j}+1},\cdots,y_k)$, it is able to find all solutions for x in polynomial time. By checking $[x]_1=X$, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ successfully finds a correct solution of the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$. □

By combining with Lemmas 1, 2, and Theorem 11, we are able to obtain an information theoretic lower bound for the GMCDH as follows.

Theorem 12 (Generic Hardness of GMCDH)**.** *The generalized matrix computational Diffie-Hellman problem in Definition 22 is $(t^2/p,t)$-hard in the generic group model if the conditions (1)–(4) of Theorem 11 simultaneously hold.*

7 Reduction for Matrix Kernel Diffie-Hellman Problem

In this section, we show generic and tight reductions from the bilinear discrete logarithm (BDL) problem in $\mathbb{G}_1$ to the matrix kernel Diffie-Hellman (MKDH) problem in an *algebraic bilinear group model specific to the MKDH* which we define in Section 7.1. In this section, we only provide a master theorem in Section 7.2 since it is much simpler than other reductions in previous sections. The simplicity stems from the fact that G_2 is the only group element in $\mathbb{G}_2$ that an MKDH adversary receives.

7.1 Algebraic Group Model for Matrix Kernel Diffie-Hellman Problem

In advance of the reduction, we define an algebraic algorithm for MKDH problems. We define the notion to be compatible with MKDH problems so that the only group element in $\mathbb{G}_2$ that the algorithm receives is a generator G_2 and the outputs are in $\mathbb{G}_2$. The definition is almost the same as Definition 29 in the sense that the algebraic algorithm is able to output only group elements which are derived from group additions in $\mathbb{G}_2$ of the only given element G_2. Furthermore, the algebraic algorithm should also output a *representation* which indicates how output group elements are calculated. Here, the algebraic algorithm outputs a discrete logarithm of the output as a representation. We formally provide a definition as follows.

Definition 32 (Algebraic Algorithm for Matrix CDH)**.** *An algorithm $\mathsf{A}_{\mathsf{alg}}$ executed in an algebraic security game $\mathbf{G}_{\mathcal{G}}$ in an asymmetric bilinear group $\mathcal{G} := (\mathbb{G}, \mathbb{G}_T, G_1, G_2, e, p)$ is called* algebraic *if for all group elements $Z \in \mathbb{G}_2$ that $\mathsf{A}_{\mathsf{alg}}$ outputs, it additionally returns the representation of Z with respect to given group elements. Specifically, $\mathsf{A}_{\mathsf{alg}}$ must also return $z \in \mathbb{Z}_p$ such that $Z = zG_2$. We use $[Z]_z$ to denote such an output.*

To be precise, the definition captures not only the MKDH problem but also any computational problem all of whose inputs and outputs are in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

7.2 Master Theorem for Matrix Kernel Diffie-Hellman Problem

In this subsection, we provide the following master theorem for the GMKDH problem.

Theorem 13 (Master Theorem for the MKDH Problem)**.** $\mathbf{BDL}_{\mathcal{G}} \Rightarrow_{\mathsf{alg}} \mathbf{GMKDH}_{\mathcal{G}}$ *holds when the following conditions are simultaneouslyl satisfied: There is at least one index $\hat{\ell} \in \{1, 2, \ldots, m\}$ such that*

- $\deg_{x_{\hat{\ell}}} f_{i,j}(x_1, \ldots, x_m) \in \{0, 1\}$ *for all* $(i, j) \in \{1, 2, \ldots, k+1\} \times \{1, 2, \ldots, k\}$.
- *There are no integer vectors* $\mathbf{v} = (v_1, \ldots, v_{k+1}) \in \mathbb{Z}_p^{k+1} \setminus \{\mathbf{0}_{k+1}\}$ *that satisfy* $\sum_{i=1}^{k+1} v_i f_{i,j}(x_1, \ldots, x_m) = 0 \mod p$ *for all* $j \in \{1, 2, \ldots, k\}$.

Then, we show a proof of Theorem 13. 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 GMKDH 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 $\mathbf{GMKDH}_{\mathcal{G}}$ only once and construct an algebraic adversary $\mathsf{B}_{\mathsf{alg}} := \mathsf{R}_{\mathsf{gen}}^{\mathsf{A}_{\mathsf{alg}}}$ on the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$.

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ is given a group description $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e, p)$ and an instance of the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$, i.e., $X := [x]_1 \in \mathbb{G}$ for an unknown $x \in \mathbb{Z}_p$. Thanks to the condition (1), we use the notation

$$f_{i,j}(x_1, \ldots, x_m) = f_{i,j}^{\mathsf{L}}(x_1, \ldots, x_{\hat{\ell}-1}, x_{\hat{\ell}+1}, \ldots, x_m)x_{\hat{\ell}} + f_{i,j}^{\mathsf{R}}(x_1, \ldots, x_{\hat{\ell}-1}, x_{\hat{\ell}+1}, \ldots, x_m)$$

for all $(i, j) \in \{1, 2, \ldots, k+1\} \times \{1, 2, \ldots, k\}$. Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ creates an instance of the $\mathbf{GMKDH}_{\mathcal{G}}$ as follows: Pick random $(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \cdots, r_m) \xleftarrow{\$} \mathbb{Z}_p^{m-1}$. If there are integer vectors $\mathbf{z} = (z_1, \ldots, z_{k+1}) \in \mathbb{Z}_p^{k+1}$ that satisfy $\sum_{i=1}^{k+1} z_i f_{i,j}^{\mathsf{L}}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) =$

$\sum_{i=1}^{k+1} z_i f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) = 0 \mod p$ for all $j \in \{1, 2, \ldots, k\}$, then resample the values. Otherwise, compute

$$\begin{aligned} X_i &= f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \cdot X + f^{\mathsf{R}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \cdot G_1 \\ &= [f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \cdot x + f^{\mathsf{R}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m)]_1 \in \mathbb{G}_1 \end{aligned}$$

for all $(i, j) \in \{1, 2, \ldots, k+1\} \times \{1, 2, \ldots, k\}$ by implicitly setting

$$(x_1, \ldots, x_{\hat{\ell}-1}, x_{\hat{\ell}}, x_{\hat{\ell}+1}, \cdots, x_m) = (r_1, \ldots, r_{\hat{\ell}-1}, x, r_{\hat{\ell}+1}, \cdots, r_m).$$

Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ gives a bilinear group description $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e, p)$ and group elements $(X_{i,j})_{(i,j)\in\{1,2,\ldots,k+1\}\times\{1,2,\ldots,k\}} \in \mathbb{G}_1^{k(k+1)}$ to $\mathsf{A}_{\mathsf{alg}}$. Observe that a set of group elements is a valid GMKDH instance 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 $\mathsf{Adv}^{\mathbf{GMKDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$ and a running time $\mathsf{Time}^{\mathbf{GMKDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$.

The reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ gives a group description $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e, p)$ and group elements $(X_{i,j} := [f_{i,j}(x_1, \ldots, x_m)]_1)_{(i,j)\in\{1,2,\ldots,k+1\}\times\{1,2,\ldots,k\}} \in \mathbb{G}_1^{k(k+1)}$ to $\mathsf{A}_{\mathsf{alg}}$. The set of group elements is a valid GMKDH instance as observed above. Hence, an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ outputs a correct solution $[\vec{Z}]_{\vec{z}}$ with an advantage $\mathsf{Adv}^{\mathbf{GMKDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$ and a running time $\mathsf{Time}^{\mathbf{GMKDH}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$.

Next, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ uses $[Z_1]_{z_1}, \ldots, [Z_{k+1}]_{z_{k+1}}$ output by an algebraic adversary $\mathsf{A}_{\mathsf{alg}}$ on the $\mathbf{GMKDH}_{\mathcal{G}}$ and computes a solution of the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$. Assume the output is a correct solution of the GMKDH, i.e.,

$$\sum_{i=1}^{k+1} z_i \left(f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m)x + f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \right) = 0 \mod p.$$

It holds with probability $\mathsf{Adv}^{\mathbf{GMKDH}_{\mathcal{G}}}_{\mathcal{G},\mathsf{A}_{\mathsf{alg}}}$. Then, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ finds an index $\hat{j}$ such that $\sum_{i=1}^{k+1} z_i f^{\mathsf{L}}_{i,j}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \neq 0 \mod p$ and obtains the following univariate equation modulo a prime p:

$$\begin{aligned} &\sum_{i=1}^{k+1} z_i \left(f^{\mathsf{L}}_{i,\hat{j}}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m)x + f^{\mathsf{L}}_{i,\hat{j}}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \right) \\ = &\left(\sum_{i=1}^{k+1} z_i f^{\mathsf{L}}_{i,\hat{j}}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \right) x + \left(\sum_{i=1}^{k+1} z_i f^{\mathsf{R}}_{i,\hat{j}}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \right) = 0 \mod p. \end{aligned}$$

Thanks to the above check, $\sum_{i=1}^{k+1} z_i f^{\mathsf{L}}_{i,\hat{j}}(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m) \neq 0$ holds and the polynomial is non-zero. Since the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ knows values of $(r_1, \ldots, r_{\hat{\ell}-1}, r_{\hat{\ell}+1}, \ldots, r_m, z_1, \ldots, z_{k+1})$, it is able to find all solutions for x in polynomial time. By checking $[x]_1 = X$, the reduction algorithm $\mathsf{R}_{\mathsf{gen}}$ successfully finds a correct solution of the $\mathbf{BDL}_{\mathcal{G}}$ in $\mathbb{G}_1$. □

By combining with Lemmas 1, 2, and Theorem 13, we are able to obtain an information theoretic lower bound for the GMKDH as follows.

Theorem 14 (Generic Hardness of GMKDH)**.** *The generalized matrix kernel Diffie-Hellman problem in Definition 28 is $(t^2/p, t)$-hard in the generic group model if the conditions of Theorem 13 simultaneously hold.*

8 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 CBDH 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. Furthermore, we also provide analogous results for the MCDH and MKDH problems.

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). 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.

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.

[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.

[JR14] Charanjit S. Jutla and Arnab Roy. Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 2014.

[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.

[MTT19] Taiga Mizuide, Atsushi Takayasu, and Tsuyoshi Takagi. Tight reductions for Diffie-Hellman variants in the algebraic group model. In Mitsuru Matsui, editor, *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, Proceedings*, volume 11405 of *Lecture Notes in Computer Science*, pages 169–188. Springer, 2019.

[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.