Multiple Discrete Logarithm Problems with Auxiliary Inputs

Taechan Kim

NTT Secure Platform Laboratories, Japan
taechan.kim@lab.ntt.co.jp

Abstract. Let g be an element of prime order p in an abelian group and let $\alpha_1, \ldots, \alpha_L \in \mathbb{Z}_p$ for a positive integer L. First, we show that, if g, g^{α_i}, and $g^{\alpha_i^d}$ $(i = 1, \ldots, L)$ are given for $d \mid p-1$, all the discrete logarithms α_i's can be computed probabilistically in $\widetilde{O}(\sqrt{L \cdot p/d} + \sqrt{L \cdot d})$ group exponentiations with $O(L)$ storage under the condition that $L \ll \min\{(p/d)^{1/4}, d^{1/4}\}$.

Let $f \in \mathbb{F}_p[x]$ be a polynomial of degree d and let ρ_f be the number of rational points over $\mathbb{F}_p$ on the curve determined by $f(x) - f(y) = 0$. Second, if $g, g^{\alpha_i}, g^{\alpha_i^2}, \ldots, g^{\alpha_i^d}$ are given for any $d \geq 1$, then we propose an algorithm that solves all α_i's in $\widetilde{O}(\max\{\sqrt{L \cdot p^2/\rho_f}, L \cdot d\})$ group exponentiations with $\widetilde{O}(\sqrt{L \cdot p^2/\rho_f})$ storage. In particular, we have explicit choices for a polynomial f when $d \mid p \pm 1$, that yield a running time of $\widetilde{O}(\sqrt{L \cdot p/d})$ whenever $L \leq \frac{p}{c \cdot d^3}$ for some constant c.

Keywords: Discrete Logarithm Problem; Multiple Discrete Logarithm; Birthday Problem; Cryptanalysis.

1 Introduction

Let G be a cyclic group of prime order p with a generator g. A discrete logarithm problem (DLP) aims to find the element α of $\mathbb{Z}_p$ when g and g^α are given. The DLP is a classical hard problem in computational number theory, and many encryption schemes, signatures, and key exchange protocols rely on the hardness of the DLP for their security.

In recent decades, many variants of the DLP have been introduced. These include the Weak Diffie–Hellman Problem [13], Strong Diffie–Hellman Problem [2], Bilinear Diffie–Hellman Inversion Problem [1], and Bilinear Diffie–Hellman Exponent Problem [3], and are intended to guarantee the security of many cryptosystems, such as traitor tracing [13], short signatures [2], ID-based encryption [1], and broadcast encryption [3]. These problems incorporate additional information to the original DLP problem. Although such additional information could weaken the problems, and their hardness is not well understood, these variants are widely used because they enable the construction of cryptosystems with various functionalities.

These variants can be considered as the problem of finding α when $g, g^{\alpha^{e_1}}, \ldots, g^{\alpha^{e_d}}$ are given for some $e_1, \ldots, e_d \in \mathbb{Z}$. This problem is called the discrete logarithm problem with auxiliary inputs (DLPwAI).

On the other hand, in the context of elliptic curve cryptography, because of large computational expense of generating a secure elliptic curve, a fixed curve is preferred to a random curve. One can choose a curve recommended by standards such as NIST. Then this causes an issue with the multiple DLP/DLPwAI and leads the following question. Can it be more efficient to solve them together than to solve each of instances individually when needed, if an adversary collects many instances of DLP/DLPwAI from one fixed curve?

In multiple discrete logarithm problem, an algorithm [11] computes L discrete logarithms in time $\widetilde{O}(\sqrt{L \cdot p})$ for $L \ll p^{1/4}$. Recently, it is proven that this algorithm is optimal in the sense that it requires at least $\Omega(\sqrt{L \cdot p})$ group operations to solve the multiple DLP in the generic group model [19].

On the other hand, an efficient algorithm for solving the DLPwAI is proposed by Cheon [5,6]. If g, g^{α}, and $g^{\alpha^d} \in G$ (resp. $g, g^{\alpha}, \ldots, g^{\alpha^{2d}} \in G$) are given, then one can solve the discrete logarithm $\alpha \in \mathbb{Z}_p$ in $O(\sqrt{p/d} + \sqrt{d})$ (resp. $O(\sqrt{p/d} + d)$) group operations in the case of $d \mid p-1$ (resp. $d \mid p+1$). Since solving the DLPwAI in the generic group model requires at least $\Omega(\sqrt{p/d})$ group operations [2], Cheon's algorithm achieves the lower bound complexity in the generic group model when $d \leq p^{1/2}$ (resp. $d \leq p^{1/3}$). Brown and Gallant [4] independently investigated an algorithm in the case of $d \mid p-1$.

However, as far as we know, the DLPwAI algorithm in the multi-user setting has not been investigated yet. This paper proposes an algorithm to solve the multiple DLPwAI better than $O(L \cdot \sqrt{p/d})$ group operations in the case of $d \mid p \pm 1$, where L denotes the number of the target discrete logarithms.

Our Contributions. We propose two algorithms for the multiple DLPwAI. Our first algorithm is based on Cheon's $(p-1)$-algorithm [5,6]. If g, g^{α_i}, and $g^{\alpha_i^d}$ $(i = 1, 2, \ldots, L)$ are given for $d \mid p-1$, our algorithm solves L discrete logarithms probabilistically in $\widetilde{O}(\sqrt{L \cdot p/d} + \sqrt{L \cdot d})$ group operations with storages for $O(L)$ elements whenever $L \leq \min\{c_{p/d}(p/d)^{1/4}, c_d d^{1/4}\}$ (for some constants $0 < c_{p/d}, c_d < 1$). We also show a deterministic variant of this algorithm which applies for any $L > 0$ and has the running time of $\widetilde{O}(\sqrt{L \cdot p/d} + \sqrt{L \cdot d} + L)$, although it requires as large amount of the storage as the time complexity. However, an approach based on Cheon's $(p+1)$-algorithm does not apply to improve an algorithm in multi-user setting.

Our second algorithm is based on Kim and Cheon's algorithm [10]. The algorithm basically works for any $d > 0$. Let $f(x) \in \mathbb{F}_p[x]$ be a polynomial of degree d over $\mathbb{F}_p$ and define $\rho_f := |(x, y) \in \mathbb{F}_p \times \mathbb{F}_p : f(x) = f(y)|$. If $g, g^{\alpha_i}, g^{\alpha_i^2}, \ldots, g^{\alpha_i^d}$ $(i = 1, 2, \ldots, L)$ are given, the algorithm computes all α_i's in $\widetilde{O}\big(\max\{\sqrt{L \cdot p^2/\rho_f}, L \cdot d\}\big)$ group operations with the storage for $\widetilde{O}(\sqrt{L \cdot p^2/\rho_f})$ elements.

In particular, if $L \cdot d \leq \sqrt{L \cdot p^2/\rho_f}$ (i.e. $L \leq \frac{p^2}{d^2 \cdot \rho_f}$), the time complexity is given by $\widetilde{O}(\sqrt{L \cdot p^2/\rho_f})$. Since $p \leq \rho_f \leq dp$, this value is always between $\widetilde{O}(\sqrt{L \cdot p/d})$ and $\widetilde{O}(\sqrt{L \cdot p})$. Explicitly, if $d \mid p-1$, one can choose the polynomial by $f(x) = x^d$ and in the case the complexity is given by the lower bound $\widetilde{O}(\sqrt{L \cdot p/d})$ whenever $L \leq p/d^3$. Similarly, in the case of $d \mid p+1$, if one takes the polynomial $f(x) = D_d(x, a)$, where $D_d(x, a)$ is the Dickson polynomial of degree d for some nonzero $a \in \mathbb{F}_p$, then it also has the running time of $\widetilde{O}(\sqrt{L \cdot p/d})$ for $L \lesssim p/(2d^3)$.

As far as the authors know, these two algorithms extend all existing DLPwAI-solving algorithms to the algorithms for multi-user setting.

Organization. This paper is organized as follows. In Section 2, we introduce several variants of DLP including a problem called discrete logarithm problem in the exponent (DLPX). We also show that several generic algorithms can be applied to solve the DLPX. In Section 3, we propose an algorithm solving the multiple DLPwAI based on Cheon's algorithm. In Section 4, we present another algorithm to solve the multiple DLPwAI using Kim and Cheon's algorithm. We conclude with some related open questions in Section 5.

2 Discrete Logarithm Problem and Related Problems

In this section, we introduce several problems related to the discrete logarithm problem. Throughout the paper, let $G = \langle g \rangle$ be a cyclic group of prime order p. Let $\mathbb{F}_q$ be a finite field with q elements for some prime power $q = p^r$. Let $\mathbb{Z}_N$ be the set of the residue classes of integers modulo an integer N.

- The *Discrete Logarithm Problem (DLP)* in G is: Given $g, g^\alpha \in G$, to solve $\alpha \in \mathbb{Z}_p$.
- The *Multiple Discrete Logarithm Problem (MDLP)* in G is: Given $g, g^{\alpha_1}, \ldots, g^{\alpha_L} \in G$, to solve all $\alpha_1, \ldots, \alpha_L \in \mathbb{Z}_p$.
- The $(e_1, \ldots, e_d)$*-Discrete Logarithm Problem with Auxiliary Inputs (DLPwAI)* in G is: Given $g, g^{\alpha^{e_1}}, g^{\alpha^{e_2}}, \ldots, g^{\alpha^{e_d}} \in G$, to solve $\alpha \in \mathbb{Z}_p$.
- The $(e_1, \ldots, e_d)$*-Multiple Discrete Logarithm Problem with Auxiliary Inputs (MDLPwAI)* in G is: Given $g, g^{\alpha_i^{e_1}}, g^{\alpha_i^{e_2}}, \ldots, g^{\alpha_i^{e_d}} \in G$ for $i = 1, 2, \ldots, L$, to solve $\alpha_1, \ldots, \alpha_L \in \mathbb{Z}_p$.

In the case of $(e_1, e_2, \ldots, e_d) = (1, 2, \ldots, d)$, we simply denote $(1, 2, \ldots, d)$-(M)DLPwAI by d-(M)DLPwAI.

We also introduce the problem called $\mathbb{F}_p$*-discrete logarithm problem in the exponent* ($\mathbb{F}_p$*-DLPX).*

- The $\mathbb{F}_p$*-Discrete Logarithm Problem in the Exponent* ($\mathbb{F}_p$*-DLPX)* in G is defined as follows: Let $\chi \in \mathbb{F}_p$ be an element of multiplicative order N, i.e. $N \mid p-1$. Given $g, g^{\chi^n} \in G$ and $\chi \in \mathbb{F}_p$, compute $n \in \mathbb{Z}_N$.
- The $\mathbb{F}_p$*-Multiple Discrete Logarithm Problem in the Exponent* ($\mathbb{F}_p$*-MDLPX)* in G is: Given $g, g^{\chi^{n_1}}, \ldots, g^{\chi^{n_L}} \in G$ and $\chi \in \mathbb{F}_p$, to solve $n_1, \ldots, n_L \in \mathbb{Z}_N$.

In both cases, the $\mathbb{F}_p$-(M)DLPX is said to be defined over $\mathbb{Z}_N$.

Algorithm for DLPX. Observe that several DL-solving algorithms can be applied to solve the DLPX with the same complexity. For example, the baby-step-giant-step (BSGS) algorithm works as follows: Suppose that the DLPX is defined over $\mathbb{Z}_N$. Set an integer $K \approx \sqrt{N}$ and write $n = n_0 K + n_1$, where $0 \leq n_0 \leq N/K \approx \sqrt{N}$ and $0 \leq n_1 < K$. For given $g, g^{\chi^n} \in G$ and $\chi \in \mathbb{F}_p$, compute and store the elements $g^{\chi^{i \cdot K}} = \left(g^{\chi^{(i-1)\cdot K}}\right)^{\chi^K}$ for all $i = 0, 1, \ldots, N/K$. Then compute $\left(g^{\chi^n}\right)^{\chi^{-j}}$ for all $j = 0, 1, \ldots, K-1$ and find a match between the stored elements. Then the discrete logarithm is given by $n = iK + j$ for the indices i and j corresponding to the match. It costs $O(\sqrt{N})$ group exponentiations by elements in $\mathbb{F}_p$ and $O(\sqrt{N})$ storage.

In a similar fashion, it is easy to check that the Pollard's lambda algorithm [15] also applies to solve the DLPX. It takes $O(\sqrt{N})$ group operations to solve the problem with small amount of storage. Also, check that the other algorithms such as Pohlig-Hellman algorithm [14] or the distinguished point method of Pollard's lambda algorithm [17] apply to solve the DLPX. The above observation was a main idea to solve the DLPwAI in [5,6].

3 Multiple DLPwAI: Cheon's algorithm

In this section, we present an algorithm of solving the $(1, d)$-MDLPwAI based on Cheon's algorithm [5,6] when $d \mid p - 1$.

Workflow of this section. Description of our algorithm is presented as follows. First, we recall how Cheon's algorithm solves the DLPwAI. In Section 3.1, we observed that the DLPwAI actually reduces to the DLPX (defined in Section 2) by Cheon's algorithm. It is, then, easy to check that to solve the MDLPwAI reduces to solve the MDLPX. So, we present an algorithm to solve the MDLPX in Section 3.2. Combined with the above results, we present an algorithm to solve the MDLPwAI in Section 3.3.

3.1 Reduction of DLPwAI to DLP in the exponent using Cheon's algorithm

We briefly remind Cheon's algorithm in the case of $d \mid p-1$. The algorithm solves $(1, d)$-DLPwAI. Let g, g^{α}, and g^{α^d} be given. Let ζ be a primitive element of $\mathbb{F}_p$ and $H = \langle \xi \rangle = \langle \zeta^d \rangle$ be a subgroup of $\mathbb{F}_p^*$ of order $\frac{p-1}{d}$. Since $\alpha^d \in H$, we have $\alpha^d = \xi^k$ for some $k \in \mathbb{Z}_{(p-1)/d}$. Our first task is to find such k. This is equivalent to solve the $\mathbb{F}_p$-DLPX defined over $\mathbb{Z}_{(p-1)/d}$, that is, to compute $k \in \mathbb{Z}_{(p-1)/d}$ for given $g, g^{\xi^k} \in G$ and $\xi \in \mathbb{F}_p$. Note that $g^{\xi^k} = g^{\alpha^d}$ is given from an instance of the DLPwAI and we know the value of ξ, since a primitive element in $\mathbb{F}_p$ can be efficiently found. As mentioned before, solving the DLPX over $\mathbb{Z}_{(p-1)/d}$ takes $O\left(\sqrt{p/d}\right)$ group exponentiations using BSGS algorithm or Pollard's lambda algorithm.

Continuously, if we write $\alpha \in \mathbb{F}_p$ as $\alpha = \zeta^\ell$, then since $\alpha^d = \zeta^{d\ell} = \zeta^{dk} = \xi^k$, it satisfies $\ell \equiv k \pmod{(p-1)/d}$, i.e. $\alpha\zeta^{-k} = (\zeta^{\frac{p-1}{d}})^m$ for some $m \in \mathbb{Z}_d$. Now we know the value of k, it remains to recover m. This is equivalent to solve $\mathbb{F}_p$-DLPX over $\mathbb{Z}_d$, that is, to solve $m \in \mathbb{Z}_d$ given the elements $g, g^{\mu^m} = (g^\alpha)^{\zeta^{-k}} \in G$ and $\mu \in \mathbb{F}_p$, where $\mu = \zeta^{\frac{p-1}{d}}$ is known. This step costs $O(\sqrt{d})$ group exponentiations. Overall, Cheon's $(p-1)$ algorithm reduces of solving two instances of DLP in the exponent with complexity $O(\sqrt{p/d} + \sqrt{d})$.

3.2 Algorithm for Multiple DLP in the Exponent

In this section, we describe an algorithm to solve *L-multiple DLP in the exponent*: Let L be a positive integer. Let χ be an element in $\mathbb{F}_p$ of multiplicative order N. The problem is to solve all $k_i \in \mathbb{Z}_N$ for given $g, y_1 := g^{\chi^{k_1}}, \ldots, y_L := g^{\chi^{k_L}}$ and χ.

We use Pollard's lambda-like algorithm. Define pseudo-random walk f from $y := g^{\chi^k}$ $(k \in \mathbb{Z}_N)$ as follows. For an integer I, define a pseudo-random function $\iota : \{g^{\chi^n} : n \in \mathbb{Z}_N\} \to \{1, 2, \ldots, I\}$ and set $S := \{\chi^{s_1}, \ldots, \chi^{s_I}\}$ for some random integers s_i. For $y = g^{\chi^k}$, a pseudo-random walk f is defined by $f : y \mapsto y^{\chi^{s_{\iota(y)}}} = g^{\chi^{k+s_{\iota(y)}}}$.

Notice that Pollard's rho-like algorithm does not apply to solve the DLPX.[1] For instance, it seems hard to compute $g^{\chi^{2k}}$ from g^{χ^k} for unknown k if the Diffie-Hellman assumption holds in the group G. This is why we take Pollard's lambda-like approach.

The proposed algorithm is basically the same with the method by Kuhn and Struik [11]. It uses the distinguished point method of Pollard's rho (lambda) method [17]. Applying their method in the case of the DLPX, we describe the algorithm as follows.

Step 1. For $y_0 := g^{\chi^{k_0}}$ for $k_0 = N - 1$, compute the following chain until it reaches to a distinguished point d_0.

$$C_0 : y_0 \mapsto f(y_0) \mapsto f(f(y_0)) \mapsto \cdots \mapsto d_0.$$

Step 2. For $y_1 = g^{\chi^{k_1}}$, compute a chain until a distinguished point d_1 found.

$$C_1 : y_1 \mapsto f(y_1) \mapsto f(f(y_1)) \mapsto \cdots \mapsto d_1.$$

If we have a collision $d_1 = d_0$, then it reveals a discrete logarithm k_1. Otherwise, set $y_1' = y_1 \cdot g^{\chi^z}$ for known z and use it as a new starting point to compute a new chain to obtain a collision.

Step 3. Once we have found the discrete logarithm $k_1, \ldots, k_i$, then one iteratively computes the next discrete logarithm k_{i+1} as follows: Compute a chain as Step 2 with a starting point y_{i+1} until a distinguished point d_{i+1} is found. Then try to find a collision $d_{i+1} = d_j$ for some $1 \le j \le i$. It reveals the

[1] In the paper [16], they indeed consider Pollard's lambda algorithm rather than rho algorithm.

discrete logarithm of y_{i+1}. If it fails, compute a chain again with a new randomized starting point $y'_{i+1} = y_{i+1} \cdot g^{\chi^{z'}}$ for known z'.

By the analysis in [11], this algorithm has a running time of $\widetilde{O}(\sqrt{L \cdot N})$ operations for $L \leq c_N N^{1/4}$ (where $0 < c_N < 1$ is some constant depending on N) with storage for $O(L)$ elements of the distinguished points.

Remark 1. If we allow large amount of storage, then we have a deterministic algorithm solving the DLPX based on the BSGS method.[2] It works for any $L \geq 0$ as follows. First, choose an integer $K = \lceil \sqrt{N/L} \rceil$ and compute $g^{\chi^{K \cdot t}} = \left(g^{\chi^{K \cdot (t-1)}}\right)^{\chi^K}$ for all $t \leq \sqrt{L \cdot N}$ using $O(\sqrt{L \cdot N})$ group exponentiations and store all of the elements. Then, for each $i = 1, 2, \ldots, L$, compute $g^{\chi^{k_i - s}} = \left(g^{\chi^{k_i}}\right)^{\chi^{-s}}$ for all $s \leq \sqrt{N/L}$ and find a collision with the stored elements. It takes $O(L \cdot \sqrt{N/L})$ operations for all. If one has a collision, then we have $k_i = s + t \cdot K$ for the indices s and t corresponding to the collision.

Remark 2. There is a recent paper by [7] that claims that the MDLP can be solved in $\widetilde{O}(\sqrt{L \cdot N})$ for any L with small amount of storage. However, their analysis (Section 2, [7]) seems somewhat questionable.

In their analysis, they essentially assumed that a collision occurs independently from each different chains. The pseudo-random function, however, once it has been fixed, it becomes deterministic and not random. For example, assume that we have a collision between two chains, say C_1 and C_2. If a new chain C_3 also collides with C_1, then it deterministically collides with C_2, too. This contradicts with independency assumption. The event that the chain C_3 connects to the chain C_2 should be independent whether C_3 is connected to C_1 or not. This kind of heuristic might be of no problem when L is much smaller than compared to N. However, this is not the case for large L.

Several literatures focus on this *rigour* of pseudo-random function used in Pollard's algorithm. For further details on this, refer to [9].

3.3 Solving Multiple DLPwAI using Cheon's algorithm

Combined with the results from Section 3.1 and Section 3.2, we propose an algorithm solving the $(1, d)$-MDLPwAI in the case of $d \mid p - 1$. In Appendix A, we explain that Cheon's $(p + 1)$-algorithm does not help to solve the MDLPwAI in the case of $d \mid p + 1$.

Theorem 1 (Algorithm for $(1, d)$-MDLPwAI, $d \mid p - 1$). *Let the notations as above. Let $\alpha_1, \ldots, \alpha_L$ be randomly chosen elements from $\mathbb{Z}_p$. Assume that $d \mid p - 1$. For $L \leq \min\{c_{p/d}(p/d)^{1/4}, c_d d^{1/4}\}$ (where $0 < c_{p/d}, c_d < 1$ are some constants on p/d and d respectively), given the elements g, g^{α_i} and $g^{\alpha_i^d}$ for $i = 1, 2, \ldots, L$, we have an algorithm that computes α_i's in $\widetilde{O}(\sqrt{L \cdot p/d} + \sqrt{L \cdot d})$ group exponentiations with storage for $O(L)$ elements in the set of the distinguished points.*

[2] The proof is contributed by Mehdi Tibouchi.

Proof. Similarly as in Section 3.1, let $H = \langle \xi \rangle = \langle \zeta^d \rangle \subset G$ for a primitive element $\zeta \in \mathbb{F}_p$. Since $\alpha_i^d \in H$, we have $\alpha_i^d = \xi^{k_i}$ for some $k_1, \ldots, k_L$, where $k_i \in \mathbb{Z}_{(p-1)/d}$, and if we write $\alpha_i = \zeta^{\ell_i}$, then we have $\alpha_i \zeta^{-k_i} = \mu^{m_i}$ for $m_i \in \mathbb{Z}_d$. Thus the problem reduces of solving two multiple DLP in the exponent with instances $g^\xi, g^{\xi^{k_1}} = g^{\alpha_1^d}, \ldots, g^{\xi^{k_L}} = g^{\alpha_L^d}$ and $g^\mu, g^{\mu^{m_1}} = (g^{\alpha_1})^{\zeta^{-k_1}}, \ldots, g^{\mu^{m_L}} = (g^{\alpha_L})^{\zeta^{-k_L}}$, where ξ and μ are known. We compute α_i's as follows:

1. Given $g^\xi, g^{\alpha_1^d} = g^{\xi^{k_1}}, \ldots, g^{\alpha_L^d} = g^{\xi^{k_L}}$ for $k_i \in \mathbb{Z}_{(p-1)/d}$, compute k_i's using the algorithm in Section 3.2. It takes time $\widetilde{O}(\sqrt{L \cdot p/d})$ with storage for $O(L)$ elements.
2. Given $g^{\alpha_1}, \ldots, g^{\alpha_L}$ and $k_1, \ldots, k_L$, compute $\zeta^{-k_1}, \ldots, \zeta^{-k_L}$ in $O(L)$ exponentiations in $\mathbb{F}_p$ and compute
$$g^{\mu^{m_1}} = (g^{\alpha_1})^{\zeta^{-k_1}}, \ldots, g^{\mu^{m_L}} = (g^{\alpha_L})^{\zeta^{-k_L}}$$
in $O(L)$ exponentiations in G.
3. Compute $m_1, \ldots, m_L \in \mathbb{Z}_d$ from $g^{\mu^{m_1}}, \ldots, g^{\mu^{m_L}}$ using the algorithm in Section 3.2. It takes time $\widetilde{O}(\sqrt{L \cdot d})$ with storage for $O(L)$ elements.

The overall complexity is given by $\widetilde{O}(\sqrt{L \cdot p/d} + \sqrt{L \cdot d} + L)$. Since $L \leq \min\{p/d, d\}$ by the assumption, i.e. $L \leq \min\{\sqrt{L \cdot p/d}, \sqrt{L \cdot d}\}$, it is equivalent to $\widetilde{O}(\sqrt{L \cdot p/d} + \sqrt{L \cdot d})$. □

Remark 3. Note that we can replace the algorithm to solve the MDLPX used in Step 1 and Step 3 with any algorithm solving the MDLPX. In that case, the complexity solving the MDLPwAI totally depends on that of the algorithm solving the MDLPX. For example, if we use the BSGS method described in Remark 1, then the proposed algorithm solves the MDLPwAI for any L in time complexity $O(\sqrt{L \cdot p/d} + \sqrt{L \cdot d} + L)$ with the same amount of storage.

4 Multiple DLPwAI: Kim and Cheon's algorithm

In this section, we propose an approach to solve the d-MDLPwAI. The idea is basically based on Kim and Cheon's algorithm [10]. To analyze the complexity, we also need some discussion on non-uniform birthday problem.

4.1 Description of Algorithm

Let $G = \langle g \rangle$ be a group of prime order p. For $i = 1, 2, \ldots, L$, let $g, g^{\alpha_i}, \ldots, g^{\alpha_i^d}$ be given. We choose a polynomial $f(x) \in \mathbb{F}_p[x]$ of degree d and fix a positive integer ℓ which will be defined later. The proposed algorithm is described as follows:

Step 1. For each i, given $g, g^{\alpha_i}, \ldots, g^{\alpha_i^d}$ and $f(x)$, we compute and store a constant number of sets each of which is of form
$$S_i := \{g^{f(r_{i,1}\alpha_i)}, \ldots, g^{f(r_{i,\ell}\alpha_i)}\},$$
where $r_{i,j}$'s are randomly chosen from $\mathbb{F}_p$.

Step 2. We also compute and store a constant number of sets each of which consists of

$$S_0 := \{g^{f(s_1)}, \ldots, g^{f(s_\ell)}\},$$

where s_k's are known random values from $\mathbb{F}_p$.

Step 3. We construct a random graph with L vertices: we add an edge between vertices i and j, if S_i and S_j collide.

Step 4. If $f(r_{i,j}\alpha_i) = f(s_k)$ for some i, j and k, then α_i is one of d roots of the equation of degree d in variable α_i:

$$f(r_{i,j}\alpha_i) - f(s_k) = 0.$$

Step 5. If $f(r_{i,j}\alpha_i) = f(r_{i',j'}\alpha_{i'})$, for some i, j, i' and j', where α_i is known, then $\alpha_{i'}$ is one of d roots of the following equation of degree d in variable $\alpha_{i'}$:

$$\widetilde{f}(\alpha_{i'}) := f(r_{i,j}\alpha_i) - f(r_{i',j'}\alpha_{i'}) = 0.$$

We recover all α_i's when they are connected into a component with known discrete logs. In the next subsection, we analyze the complexity of the proposed algorithm more precisely.

4.2 Complexity Analysis

We analyze the complexity of the proposed algorithm.

Theorem 2 (Algorithm for d-MDLPwAI). *Let the notations as above. Let $f(x)$ be a polynomial of degree d over $\mathbb{F}_p$. Define $\rho_f := |\{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p : f(x) = f(y)\}|$. Given $g, g^{\alpha_i}, \ldots, g^{\alpha_i^d}$ for $i = 1, 2, \ldots, L$, we have an algorithm that computes all α_i's in $\widetilde{O}(\max\{\sqrt{L \cdot p^2/\rho_f}, L \cdot d\})$ group exponentiations with storage for $\widetilde{O}(\sqrt{L \cdot p^2/\rho_f})$ elements in G.*

Proof. Consider the complexity of each step in the proposed algorithm. Throughout the paper, we denote $M(d)$ by the time complexity multiplying two polynomials of degree d over $\mathbb{F}_p$ (typically, we will take $M(d) = O(d \log d \log\log d)$ using the Schönhage-Strassen method).

In Step 2, we compute $f(s_1), \ldots, f(s_\ell)$ using fast multipoint evaluation method. It takes $O(\ell/d \cdot M(d) \log d) = O(\ell \log^2 d \log\log d)$ operations in $\mathbb{F}_p$ if $\ell \geq d$. Otherwise, the cost is bounded by $O(M(d) \log d) = O(d \log^2 d \log\log d)$ operations in $\mathbb{F}_p$. Then compute $g^{f(s_1)}, \ldots, g^{f(s_\ell)}$ in $O(\ell)$ exponentiations in G.

In Step 1, we use fast multipoint evaluation method in the exponent as described in [10, Theorem 2.1], which is the following: given $g^{F_0}, \ldots, g^{F_d}$, where F_i is the coefficient of x^i of a polynomial $F(x) \in \mathbb{F}_p[x]$, and given random elements $r_1, \ldots, r_d \in \mathbb{F}_p$, it computes $g^{F(r_1)}, \ldots, g^{F(r_d)}$ in $O(M(d) \log d)$ operations in G.

In our case, for given $g, g^{\alpha_i}, \ldots, g^{\alpha_i^d}$ and $f(x) = a_0 + \cdots + a_d x^d$, we set $f_i(x) := f(\alpha_i x) = a_0 + (a_1\alpha_i)x + \cdots + (a_d\alpha_i^d)x^d$ and compute $g^{a_0}, (g^{\alpha_i})^{a_1}, \ldots, (g^{\alpha_i^d})^{a_d}$ in $O(d)$ exponentiations in G for each i. It totally costs $O(L \cdot d)$ exponentiations

for all $i = 1, \ldots, L$. Applying Theorem 2.1 in [10] to each polynomial $f_i(x)$, if $\ell \geq d$, we compute

$$S_i = \{g^{f_i(r_{i,1})}, \ldots, g^{f_i(r_{i,\ell})}\} = \{g^{f(r_{i,1}\alpha_i)}, \ldots, g^{f(r_{i,\ell}\alpha_i)}\}$$

in $O(\ell/d \cdot M(d) \log d)$ operations in G for each i. It costs $O(L \cdot \ell \log^2 d \log\log d)$ operations overall for all $i = 1, \ldots, L$. Otherwise, if $\ell \leq d$, then this step costs $O(L \cdot d \log^d \log\log d)$ operations.

In Step 4 and Step 5, the cost takes $O(M(d) \log d \log(dp))$ field operations on average [18] to compute roots of equation of degree d over $\mathbb{F}_p$. For each equation, we need to find α_i among at most d possible candidates. It takes $O(d)$ operations. These steps need to be done L times since we have L equations to be solved.

Consequently, to recover all α_i's, it takes overall $\widetilde{O}(\max\{L \cdot \ell, L \cdot d\})$ operations with $O(L \cdot \ell)$ storage. Now it remains to determine the value of ℓ. To this end, we need to clarify the probability of a collision between S_i and S_j (for $i \neq j$) in Step 3. It leads us to consider non-uniform birthday problem of two types. We will discuss on details for this in Appendix B.

We heuristically assume that the probability of a collision between S_i and S_j in Step 3 is equiprobable for any $i \neq j$ and we denote this probability by ω.[3] By Corollary 1 in Appendix B, the probability is given by $\omega = \Theta(\ell^2 \cdot \rho_f / p^2)$ for large p. Then the expected number of edges in the graph in Step 3 will be $\binom{L}{2} \cdot \omega \approx \frac{L^2\omega}{2} \approx \frac{L^2\ell^2}{2} \cdot \frac{\rho_f}{p^2}$. We require this value to be larger than $2L \ln L$ to connect all connected components in the graph (see [7]), i.e.

$$\ell \geq 2\sqrt{\frac{p^2}{\rho_f} \cdot \frac{\ln L}{L}}.$$

If we take $\ell = 2\sqrt{\frac{p^2}{\rho_f} \cdot \frac{\ln L}{L}}$, the overall time complexity becomes (without log terms) $\widetilde{O}(\max\{L \cdot \ell, L \cdot d\}) = \widetilde{O}\left(\max\left\{\sqrt{L \cdot p^2/\rho_f}, L \cdot d\right\}\right)$ with storage for $\widetilde{O}(L \cdot \ell) = \widetilde{O}\left(\sqrt{L \cdot p^2/\rho_f}\right)$ elements in G. □

Remark 4. In general, the computation of ρ_f seems relatively not so obvious. However, for some functions f which are useful for our purpose, it can be efficiently computable. See Section 4.3.

If $L \leq \frac{p^2}{d^2 \cdot \rho_f}$, then the time complexity of the algorithm is given by $\widetilde{O}\left(\sqrt{L \cdot p^2/\rho_f}\right)$. Note that this value is always between $\widetilde{O}\left(\sqrt{L \cdot \frac{p}{d}}\right)$ and $\widetilde{O}(\sqrt{L \cdot p})$. In the next subsection, we observe that one can find polynomials f with $\rho_f \approx C \cdot dp$ for some constant C in the case of $d \mid p \pm 1$. In such cases, the proposed algorithm has a running time of $\widetilde{O}(\sqrt{L \cdot p/d})$ whenever $L \leq \frac{p}{C \cdot d^3}$.

It should be compared that application of Cheon's $(p+1)$-algorithm failed to achieve the lower bound complexity $\widetilde{O}(\sqrt{L \cdot p/d})$ in the case of $d \mid p+1$ (see Appendix A).

[3] The assumption is reasonable, since every exponents of the elements in S_i's are randomly chosen from $\mathbb{F}_p$, i.e. the sets S_i's are independent from each other. Observe that this does not conflict with Remark 2.

4.3 Explicit Choices of Polynomials for Efficient Algorithms in the Case of $d \mid p \pm 1$

For efficiency of the algorithm, we require a polynomial $f(x)$ with large ρ_f. In particular, ρ_f becomes larger as the map $x \mapsto f(x)$, restricted on $\mathbb{F}_p$ or a large subset of $\mathbb{F}_p$, has a smaller value set. See the examples below. For details on choices of these polynomials, refer to [10].

$d \mid p-1$ Case. Let $f(x) = x^d$. Then the map by f is d-to-1 except at $x = 0$. Then we have $\rho_f = 1 + d(p-1) \approx dp$. In this case, the complexity of our algorithm becomes $\widetilde{O}\big(\sqrt{L \cdot p/d}\big)$ for $L \leq p/d^3$.

$d \mid p+1$ Case. Let $f(x) = D_d(x, a)$ be the Dickson polynomial for a nonzero element $a \in \mathbb{F}_p$, where

$$D_d(x,a) = \sum_{k=0}^{\lfloor d/2 \rfloor} \frac{d}{d-k} \binom{d-k}{k} (-a)^k x^{d-2k}.$$

If $d \mid p+1$, then by [8,12], we have $\rho_f = \frac{(d+1)p}{2} + O(d^2) \approx \frac{dp}{2}$. In this case, our algorithm has the complexity of $\widetilde{O}(\sqrt{L \cdot p/d})$ for $L \lesssim p/(2d^3)$.

5 Conclusion

In this paper, we proposed algorithms for the MDLPwAI based on two different approaches. These algorithms cover all extensions of existing DLPwAI-solving algorithms, since, up to our knowledge, there are only two (efficient) approaches solving the DLPwAI: Cheon's algorithm and Kim and Cheon's algorithm.

Our analysis shows that our algorithms have the best running time of either $\widetilde{O}(\max\{\sqrt{L \cdot p/d}, \sqrt{L \cdot d}\})$ when $d \mid p-1$, or $\widetilde{O}(\max\{\sqrt{L \cdot p/d}, L \cdot d\})$ when $d \mid p+1$. It shows that the choice of the prime p should be chosen carefully so that both of $p+1$ and $p-1$ have no small divisors. Readers might refer to [5,6] for careful choices of such prime p.

However, our second algorithm is based on some heuristics and requires relatively large amount of memory. Thus, it would be a challenging question either to reduce the storage requirement in the algorithm, or to make the algorithm more rigorous.

It would be also interesting to determine the lower bound complexity in the generic group model for solving the multiple DLPwAI. A very recent result [19] showed that at least $\Omega(\sqrt{L \cdot p})$ group operations are required to solve the L multiple DLP in the generic group model. Recall that the generic lower bound for the DLPwAI is $\Omega(\sqrt{p/d})$. Then it is natural to ask the following questions. What is the lower bound complexity in the generic group model to solve the multiple DLPwAI? Do we need at least $\Omega(\sqrt{L \cdot p/d})$ operations for solving the multiple DLPwAI?

A A Failed Approach for MDLPwAI when $d \mid p+1$

$\mathbb{F}_{p^2}$-Discrete Logarithm Problem in the Exponent To define $\mathbb{F}_{p^2}$-(M)DLPX, we introduce the following definition.[4]

Definition 1. *Let $G = \langle g \rangle$ be a group of prime order p. Let $\mathbb{F}_{p^2} = \mathbb{F}_p[\theta] = \mathbb{F}_p[x]/(x^2 - \kappa)$ for some quadratic non-residue $\kappa \in \mathbb{F}_p$. For $\gamma = \gamma_0 + \gamma_1\theta \in \mathbb{F}_{p^2}$, we define $g^\gamma = (g^{\gamma_0}, g^{\gamma_1})$ with abuse of notations. For $\mathbf{g} := (g_0, g_1) \in G \times G$, we define*

$$\mathbf{g}^\gamma = \mathbf{g}^{\gamma_0+\gamma_1\theta} = (g_0^{\gamma_0} g_1^{\kappa\gamma_1}, g_0^{\gamma_1} g_1^{\gamma_0}), \text{ where } \theta^2 = \kappa.$$

One can readily check that $(g^\gamma)^\delta = (g^{\gamma_0}, g^{\gamma_1})^\delta = (g^{\gamma_0\delta_0+\kappa\gamma_1\delta_1}, g^{\gamma_0\delta_1+\gamma_1\delta_0}) = g^{\gamma\delta}$, where $\delta = \delta_0 + \delta_1\theta$. Now we define $\mathbb{F}_{p^2}$-(M)DLPX.

- The *$\mathbb{F}_{p^2}$-Discrete Logarithm Problem in the Exponent ($\mathbb{F}_{p^2}$-DLPX)* in G is defined as follows: Let $\chi \in \mathbb{F}_{p^2}$ be an element of multiplicative order N, i.e. $N \mid p^2 - 1$. Given $g \in G$ and $g^{\chi^n} \in G \times G$ and $\chi \in \mathbb{F}_{p^2}$, compute $n \in \mathbb{Z}_N$.
- The *$\mathbb{F}_{p^2}$-Multiple Discrete Logarithm Problem in the Exponent ($\mathbb{F}_{p^2}$-MDLPX)* in G is: Given $g \in G$, $g^{\chi^{n_1}}, \ldots, g^{\chi^{n_L}} \in G \times G$ and $\chi \in \mathbb{F}_{p^2}$, to solve $n_1, \ldots, n_L \in \mathbb{Z}_N$. In both cases, the $\mathbb{F}_{p^2}$-(M)DLPX is said to be defined over $\mathbb{Z}_N$.

Observe that generic approaches to solve the (M)DLPX described in Section 2 and Section 3.2 also apply to solve the $\mathbb{F}_{p^2}$-(M)DLPX.

A failed approach when $d \mid p+1$ We consider the MDLPwAI in the case of $d \mid p+1$. Recall Cheon's $(p+1)$ algorithm [5,6] which solves $2d$-DLPwAI. Let $g, g^{\alpha_i}, \ldots, g^{\alpha_i^{2d}}$, for $i = 1, 2, \ldots, L$, be given. We try to solve the problem as follows: For each $i = 1, 2, \ldots, L$, let $\beta_i := (1 + \alpha_i\theta)^{p-1} \in \mathbb{F}_{p^2} = \mathbb{F}_p[\theta]$ and let $\xi \in \mathbb{F}_{p^2}$ an element of multiplicative order $(p+1)/d$. We compute $g_i := g^{(1-\kappa\alpha_i^2)^d}$ and $g_i^{\xi^{k_i}} = g_i^{\beta_i^d} := (g^{f_0(\alpha_i)}, g^{f_0(\alpha_i)})$ for the given elements $g, g^{\alpha_i}, \ldots, g^{\alpha_i^{2d}}$, where $\beta_i^d = \frac{1}{(1-\kappa\alpha_i^2)^d}\{f_0(\alpha_i) + f_1(\alpha_i)\theta\}$. The remaining task is to solve $k_1, \ldots, k_L \in \mathbb{Z}_{(p+1)/d}$. This translates to solve L instances of the $\mathbb{F}_{p^2}$-DLPX, say $(g_1, g_1^{\xi^{k_1}}), (g_2, g_2^{\xi^{k_2}}), \ldots, (g_L, g_L^{\xi^{k_L}})$. Note that, however, these L instances cannot be solved efficiently in a batch computation based on our MDLPX algorithms, since all the bases of the instances are not the same.

B Non-Uniform Birthday Problem: Girls and Boys

In this section, we consider the probability of a collision in Step 3, Section 4.1. More generally, we consider non-uniform birthday problem of two types. The main goal in this section is to prove the following theorem.

[4] This notion can be found in [5,6] when he solves DLPwAI using Pollard's lambda algorithm. We simply formalize them.

Theorem 3. *For a positive integer N and $i \in \{1, 2, \ldots, N\}$, assume that the probability of a randomly chosen element from the set $\{1, 2, \ldots, N\}$ to be i is ω_i. Let T_1 (respectively, T_2) be a set consisting of ℓ_1 (reps. ℓ_2) elements randomly chosen from $\{1, 2, \ldots, N\}$. Then the probability ω that T_1 and T_2 have an element in common satisfies*

$$\ell_1\ell_2 \cdot \sum_{i=1}^{N} \omega_i^2 \geq \omega \geq \ell_1\ell_2 \cdot \sum_{i=1}^{N} \omega_i^2 - \left(\ell_1 \cdot \binom{\ell_2}{2} + \ell_2 \cdot \binom{\ell_1}{2}\right) \cdot \sum_{i=1}^{N} \omega_i^3 + \binom{\ell_1}{2}\binom{\ell_2}{2}\left(\sum_{i=1}^{N} \omega_i^4 - \sum_{1 \leq i < j \leq N} \omega_i^2\omega_j^2\right). \tag{1}$$

Proof. For each $i \in \{1, 2, \ldots, N\}$, let $B_i^{(\ell_1,\ell_2)}$ be the event that two sets T_1 and T_2 have the element i in common. Then the probability ω that T_1 and T_2 have at least one element in common is given by

$$\omega = \Pr[B_1^{(\ell_1,\ell_2)} \cup \cdots \cup B_N^{(\ell_1,\ell_2)}].$$

From now on, we shall omit superscript in $B_i^{(\ell_1,\ell_2)}$ and simply denote it by B_i. To bound the value ω, we use Bonferroni inequality,

$$\sum_{i=1}^{N} \Pr[B_i] - \sum_{1 \leq i < j \leq N} \Pr[B_i \cap B_j] \leq \omega \leq \sum_{i=1}^{N} \Pr[B_i].$$[5]

We shall investigate bounds on $\Pr[B_i]$ and $\Pr[B_i \cap B_j]$ in the followings.

For each i, the set T_1 with ℓ_1 elements has the element i with probability $1 - (1 - \omega_i)^{\ell_1}$ and similarly for T_2. Thus both of T_1 and T_2 have the element i with probability $\Pr[B_i] = \left(1 - (1 - \omega_i)^{\ell_1}\right) \cdot \left(1 - (1 - \omega_i)^{\ell_2}\right)$. Using the inequality $1 - nx \leq (1 - x)^n \leq 1 - nx + \binom{n}{2}x^2$ for $0 \leq x \leq 1$ and $n > 1$, we have

$$\left(\ell_1\omega_i - \binom{\ell_1}{2}\omega_i^2\right) \cdot \left(\ell_2\omega_i - \binom{\ell_2}{2}\omega_i^2\right) \leq \Pr[B_i] \leq \ell_1\ell_2 \cdot \omega_i^2.$$

Furthermore, we have $\Pr[B_i \cap B_j] \leq \binom{\ell_1}{2}\binom{\ell_2}{2}\omega_i^2\omega_j^2$, since T_1 has the element i and j with probability at most $\binom{\ell_1}{2}\omega_i\omega_j$ and similarly for T_2.

[5] It is easy to check the lower bound inequality. Assume that $\Pr[B_1 \cup B_2] \geq \Pr[B_1] + \Pr[B_2] - \Pr[B_1 \cap B_2]$ (indeed the equality holds in this case). Then to see that

$$\begin{aligned}\Pr[(B_1 \cup B_2) \cup B_3] &= \Pr[B_1 \cup B_2] + \Pr[B_3] - \Pr[(B_1 \cup B_2) \cap B_3] \\ &\geq \Pr[B_1] + \Pr[B_2] + \Pr[B_3] - \Pr[B_1 \cap B_2] - \Pr[B_1 \cap B_3] - \Pr[B_2 \cap B_3],\end{aligned}$$

it is enough to check that

$$\Pr[(B_1 \cup B_2) \cap B_3] = \Pr[(B_1 \cap B_3) \cup (B_2 \cap B_3)] \leq \Pr[B_1 \cap B_3] + \Pr[B_2 \cap B_3].$$

Now apply the induction on N.

Then the upper bound for ω directly comes from the upper bound for $\Pr[B_i]$ and the lower bound comes from

$$\begin{aligned}\omega &\geq \sum_{i=1}^{N} \Pr[B_i] - \sum_{1\leq i<j\leq N} \Pr[B_i \cap B_j] \\ &\geq \sum_{i=1}^{N} \left(\ell_1\omega_i - \binom{\ell_1}{2}\omega_i^2\right) \cdot \left(\ell_2\omega_i - \binom{\ell_2}{2}\omega_i^2\right) - \sum_{i<j} \binom{\ell_1}{2}\binom{\ell_2}{2}\omega_i^2\omega_j^2.\end{aligned}$$

This concludes the proof. □

Corollary 1. *Let $W := \sum_{i=1}^{N} \omega_i^2$ in Theorem 3. If $\ell = \ell_1 = \ell_2 = O\left(\sqrt{\frac{1}{W}}\right)$ and $W \to 0$, we have*

$$\ell^2 W - \frac{(\ell^2 W)^2}{8} + O(\frac{1}{\sqrt{W}}) \leq \omega \leq \ell^2 W.$$

Proof. Evaluating $\ell = \ell_1 = \ell_2$ in the right most side of eq. (1), we have

$$\begin{aligned}\omega &\geq \ell^2 W - \ell^2(\ell-1)\sum_{i=1}^{N}\omega_i^3 + \frac{\ell^2(\ell-1)^2}{4}\left(\frac{3}{2}\sum_i \omega_i^4 - \frac{1}{2}W^2\right) \\ &\geq \ell^2 W - \ell^3 \sum_{i=1}^{N}\omega_i^3 - \frac{1}{8}(\ell^2 W)^2.\end{aligned}$$

In the first inequality, we used that $\sum_{i<j}\omega_i^2\omega_j^2 = \frac{1}{2}\left[\left(\sum_i \omega_i^2\right)^2 - \sum_i \omega_i^4\right]$. To see that $\ell^3 \sum_i \omega_i^3 \leq O\left(\frac{1}{\sqrt{W}}\right)$, it is enough to check that $\sum_i \omega_i^3 = \sum_i \omega_i^2\omega_i = \sum_i \omega_i^2 \cdot \sum_i \omega_i - \sum_{i\neq j}\omega_i^2\omega_j \leq \sum_i \omega_i^2 = W$ (recall that $\sum_i \omega_i = 1$). □

Return to our interest. Intrinsically, in our application (Section 4), we consider the intersection between two sets $T_1 := \{t_1, \ldots, t_\ell\} = \{f(r_1), \ldots, f(r_\ell)\}$ and $T_2 := \{t'_1, \ldots, t'_\ell\} = \{f(r'_1), \ldots, f(r'_\ell)\}$ for a degree d polynomial $f(x) \in \mathbb{F}_p[x]$. This can be regarded as non-uniform birthday problem described in Theorem 3 similarly as in [10]: An element $t \in T_1$ (or $t' \in T_2$) is randomly chosen from $\mathbb{F}_p$ with the probability $\frac{|f^{-1}(t)|}{p}$. Let $R_i := |\{t \in \mathbb{F}_p : |f^{-1}(t)| = i\}|$ for a non-negative integer i. We have $R_i = 0$ for $i > d$ since $\deg(f) = d$. Then we might say that an element in T_1 (or T_2) is drown by following the probability distribution (with proper rearrange)

$$(\omega_1, \ldots, \omega_p) = (\underbrace{0, \ldots, 0}_{R_0}, \underbrace{\frac{1}{p}, \ldots, \frac{1}{p}}_{R_1}, \underbrace{\frac{2}{p}, \ldots, \frac{2}{p}}_{R_2}, \cdots, \underbrace{\frac{d}{p}, \ldots, \frac{d}{p}}_{R_d}).$$

Then $W = \sum_{i=1}^{p}\omega_i^2 = \frac{\sum_{i=1}^{d} i^2 R_i}{p^2} = \frac{\rho_f}{p^2}$, where $\rho_f := |\{(x,y) \in \mathbb{F}_p \times \mathbb{F}_p : f(x) = f(y)\}|$. In our case, we usually take $\ell = 2\sqrt{\frac{p^2}{\rho_f} \cdot \frac{\ln L}{L}} = O(\sqrt{1/W})$ (see the proof

of Theorem 3), where L is the constant given by the number of the target discrete logarithms. Then, by Corollary 1, we roughly have $\ell^2 W - \frac{(\ell^2 W)^2}{8} \leq \omega \leq \ell^2 W$ for large enough p, i.e. $\omega = \Theta(\ell^2 W)$ (using $x - x^2/8 \geq (7/8)x$ for $0 \leq x \leq 1$). Consequently, this gives what we want for the analysis.

Acknowledgement. The author would like to thank Pierre-Alain Fouque, Soojin Roh, Mehdi Tibouchi, and Aaram Yun for their valuable discussion. He also would like to extend his appreciation to anonymous reviewers who further improved this paper.

References

1. D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.
2. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.
3. D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, 2005.
4. D. R. L. Brown and R. P. Gallant. The static Diffie-Hellman problem. *IACR Cryptology ePrint Archive*, 2004. http://eprint.iacr.org/2004/306.
5. J. H. Cheon. Security analysis of the strong Diffie-Hellman problem. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11. Springer, 2006.
6. J. H. Cheon. Discrete logarithm problems with auxiliary inputs. *J. Cryptology*, 23(3):457–476, 2010.
7. P. Fouque, A. Joux, and C. Mavromati. Multi-user collisions: Applications to discrete logarithm, even-mansour and PRINCE. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Computer Science*, pages 420–438. Springer, 2014.
8. J. Gomez-Calderon and D. J. Madden. Polynomials with small value set over finite fields. *Journal of Number Theory*, 28:167–188, 1988.
9. S. Kijima and R. Montenegro. Collision of random walks and a refined analysis of attacks on the discrete logarithm problem. In J. Katz, editor, *PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 127–149. Springer, 2015.
10. T. Kim and J. H. Cheon. A new approach to discrete logarithm problem with auxiliary inputs. *IACR Cryptology ePrint Archive*, 2012. http://eprint.iacr.org/2012/609.
11. F. Kuhn and R. Struik. Random walks revisited: Extensions of pollard's rho algorithm for computing multiple discrete logarithms. In S. Vaudenay and A. M. Youssef, editors, *Selected Areas in Cryptography 2001*, volume 2259 of *Lecture Notes in Computer Science*, pages 212–229. Springer, 2001.
12. D. A. Mit'kin. Polynomials with minimal set of values and the equation $f(x) = f(y)$ in a finite prime field. *Matematicheskie Zametki*, 38(1):3 –14, 1985.

13. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 85(2):481–484, 2002.
14. S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.
15. J. M. Pollard. Kangaroos, monopoly and discrete logarithms. *Journal of Cryptology*, 13(4):437–447, 2000.
16. Y. Sakemi, T. Izu, M. Takenaka, and M. Yasuda. Solving a DLP with auxiliary input with the ρ-algorithm. In S. Jung and M. Yung, editors, *WISA 2011*, volume 7115 of *Lecture Notes in Computer Science*, pages 98–108. Springer, 2011.
17. P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, 1999.
18. J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2003.
19. A. Yun. Generic hardness of the multiple discrete logarithm problem. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 817–836. Springer, 2015.