The Discrete Logarithm Problem with Auxiliary Inputs

Jung Hee Cheon, Taechan Kim and Yongsoo Song

Abstract. The Discrete Logarithm Problem (DLP) is a classical hard problem in computational number theory, and forms the basis of many cryptographic schemes. The DLP involves finding α for the given elements g and g^α of the cyclic group $G = \langle g \rangle$ of finite order n. Recently, many variants of the DLP have been used to ensure the security of pairing-based cryptosystems, such as ID-based encryption, broadcast encryption, and short signatures. These cryptosystems provide various functionalities, but their underlying problems are not well understood. A generalization of these variants of DLP, called the Discrete Logarithm Problem with Auxiliary Inputs (DLPwAI), aims to find α for some given $g, g^\alpha, \ldots, g^{\alpha^d}$.

This survey article first recalls several well-known solutions of the original DLP, and mainly focuses on recent attempts to solve the DLPwAI. Research into the DLPwAI started with Cheon's $p \pm 1$ algorithms [11] at EUROCRYPT '06, which use the embedding of the discrete logarithm into the extension of the finite field. Later, Satoh [34] and Kim et al. [24] tried to generalize Cheon's algorithm to the case of $\Phi_k(p)$ for $k \geq 3$, where $\Phi_k(\cdot)$ is the k-th cyclotomic polynomial. However, Kim et al. found that this generalization of Cheon's algorithm cannot be better than the usual square-root complexity algorithms, such as Pollard's rho algorithm, when $k \geq 3$.

We also introduce a recent result by Cheon and Kim [25] that reduces the DLPwAI to the problem of finding a polynomial of degree d with a small value set. Finally, we present a generalized version of the DLPwAI introduced by Cheon, Kim, and Song [15], with an algorithm for this problem, even when neither $p + 1$ or $p - 1$ has an appropriate small divisor.

Keywords. Discrete Logarithm Problem, Cheon's Algorithm, DLPwAI.

AMS classification. Please insert AMS Mathematics Subject Classification numbers here. See `www.ams.org/msc`.

1 Introduction

Let G be a cyclic group of finite order n with a generator g. The Discrete Logarithm Problem (DLP) aims to find the element α of $\mathbb{Z}_n$ when two elements g and $h = g^\alpha$ 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.

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2011-0018345). The third author was also supported by the NRF-2012-Global Ph.D. Fellowship Program.

In recent decades, many variants of the DLP have been introduced. These include the Weak Diffie–Hellman Problem [29], Strong Diffie–Hellman Problem [4], Bilinear Diffie–Hellman Inversion Problem [3], and Bilinear Diffie–Hellman Exponent Problem [8], and are intended to guarantee the security of many cryptosystems, such as traitor tracing [29], short signatures [4], ID-based encryption [3], and broadcast encryption [8]. 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.

The first study of the weakness of these problems was done by Cheon [11, 12]. He realized that these variants can be considered as the problem of finding α when $g, g^{\alpha}, \ldots, g^{\alpha^d}$ are given, and called this problem the Discrete Logarithm Problem with Auxiliary Inputs (DLPwAI). The DLPwAI can be solved efficiently with a time complexity of $O(\sqrt{p/d})$, when d is a small divisor of $p \pm 1$ for the prime order p of the group G. This complexity is the same as the lower bound of the complexity of solving the DLPwAI in the generic group model [35]. Since the lower bound for the original DLP is $\Omega(\sqrt{p})$ in the generic group model, Cheon's algorithm demonstrates the weakness of the DLPwAI in some cases. The algorithm for the case $p-1$ was independently proposed by Brown and Gallant [9].

The idea of Cheon's algorithm is to embed the discrete logarithm α into the finite fields $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$. Precisely, he exploits the fact that α^d can be embedded into an element of the small subgroup of $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$ when d is a divisor of $p \pm 1$. Later, Satoh [34] generalized this algorithm using the embedding of $\alpha \in \mathbb{F}_p$ into the general linear group $GL_k(\mathbb{F}_p)$. The generalization attempts to solve the problem when d is a divisor of $\Phi_k(p)$ for the k-th cyclotomic polynomial $\Phi_k(\cdot)$. However, its complexity for $k \geq 3$ was not clearly understood. Recently, Kim et al. [24] realized that Satoh's generalization is essentially the same as the embedding of $\mathbb{F}_p$ into $\mathbb{F}_{p^k}$, and clarified the complexity of the algorithm. Unfortunately, their result suggests that, in most cases, the complexity of this generalization is not faster than the current square-root complexity algorithm, such as Pollard's rho algorithm [31], for $k \geq 3$.

Each of the above algorithms uses the embedding technique of the finite field. This can be considered as the quantitative version of the reduction algorithms from the DLP into the Diffie–Hellman problem [27, 28]. In contrast, Kim and Cheon [25] proposed an algorithm to solve the DLPwAI with a polynomial mapping instead of embedding the element. Precisely, they compute two lists of $g^{f(r_i\alpha)}$ and $g^{f(s_j)}$ for random elements r_i and s_j for a polynomial f of degree d using fast multipoint evaluation, and find a collision between them. For this algorithm to be efficient, it is necessary to find a polynomial for which the curve defined by $f(x) - f(y) = 0$ has many rational points. Their algorithm shows the same asymptotic complexity as Cheon's algorithm for the cases of $p \pm 1$, but finding a good polynomial for $\Phi_k(p), k \geq 3$ remains an open problem.

A recent result on the DLPwAI was reported by Cheon, Kim, and Song [15]. They

introduced a generalization of the DLPwAI, called the GDLPwAI. This problem aims to find α for given $g^{\alpha^{e_1}}, \ldots, g^{\alpha^{e_d}}$ for some $e_1, \ldots, e_d$. In particular, they proposed a heuristic algorithm that solves the GDLPwAI, where the e_i are elements of a multiplicative subgroup of $\mathbb{Z}_{p-1}^{\times}$.

Organization This paper is organized as follows. In Section 2, we recall some well-known DL algorithms, including the baby–step giant–step (BSGS) algorithm, Pollard's rho/kangaroo algorithms, the Pohlig–Hellman algorithm, the index calculus algorithm, and the number/function field sieve. In Section 3, we introduce the DLPwAI, and describe Cheon's algorithm for cases of $p \pm 1$. Several attempts to generalize Cheon's algorithm are discussed in this section. Sections 4 and 5 explain another approach to solving the DLPwAI using polynomials with small value sets. In Section 6, we introduce the GDLPwAI, and propose an algorithm to solve this problem. Section 7 examines the implications of Cheon's algorithm, and some open problems and further work are discussed in Section 8.

2 Algorithms for the ordinary DLP

In this section, some classical algorithms for solving the original DLP are briefly introduced. Though the main topic of this manuscript is the DLPwAI, it is necessary to understand the basic algorithms for the original DLP to follow the arguments later in this paper. For more information, refer to [17].

2.1 Generic algorithms

First, consider the order of the cyclic group G to be an integer n that is not necessarily prime. Generic algorithms solve the DLP without using specific properties or representations of the base group G, so they can be applied to an arbitrary group. However, in the sense of the generic group model [35], the lower bound of the runtime of a generic algorithm is $\Omega(\sqrt{p})$ where p is the largest prime divisor of n.

The BSGS algorithm solves the DLP deterministically in $O(\sqrt{n})$ time. It computes α by making two lists of elements of G and finding a collision.

The Pohlig–Hellman algorithm solves the DLP efficiently when the order n of G has only small prime factors. For $n = \prod_i q_i^{f_i}$, the algorithm first solves $\alpha \mod q_i^{f_i}$ for each i using the BSGS algorithm, and then recovers α using the Chinese Remainder Theorem (CRT). Thus, the total complexity depends on the size of the largest prime factor of n.

While the BSGS algorithm requires $O(\sqrt{n})$ storage for the lists, Pollard's rho algorithm only requires constant storage size, although it is probabilistic. Pollard's kangaroo algorithm is also probabilistic, and determines the discrete logarithm α contained in the specific interval $[a, b]$. Its complexity is $O(\sqrt{b-a})$.

These algorithms are all generic in the sense that they work for a finite cyclic group G of any order n. Moreover, all of these algorithms have exponential time complexities.

Baby–step giant–step algorithm

The BSGS algorithm is a simple, generic and deterministic algorithm to solve the DLP. The total complexity is $O(\sqrt{n})$ operations in G, but it must also store $O(\sqrt{n})$ elements of G.

For a given generator g of G, we first construct a lookup table (baby–steps) which contains all the pairs (i, g^i) for $0 \le i < m$, and sort the pairs of table by the second component. To find the discrete logarithm α of a given element $h = g^\alpha$, we calculate $g^{-\lceil\sqrt{n}\rceil j}h$ for each $0 \le j < \lceil\sqrt{n}\rceil$ (giant–steps) and compare with the lookup table in order to identify coincidences.

If a collision $g^{-j_0}h = g^{\lceil\sqrt{n}\rceil i_0}$ occurs, then the discrete logarithm is calculated from $h = g^{j_0+\lceil\sqrt{n}\rceil i_0}$ and $\alpha = j_0 + \lceil\sqrt{n}\rceil i_0$. For any $0 \le \alpha < n$, there exist two integers satisfying $0 \le i_0, j_0 < \lceil\sqrt{n}\rceil$ and $\alpha = j_0 + \lceil\sqrt{n}\rceil i_0$. Therefore, the two lists L_1 and L_2 always have a common element.

Note that we do not need to store the whole elements of list L_2, and list L_1 may be reused to solve the DLP for another element h' of G. Moreover, the BSGS algorithm even works when the order n of G is not known by substituting the size $\lceil\sqrt{n}\rceil + 1$ of the lists for a sufficiently large integer ℓ.

The Pohlig–Hellman algorithm

If all prime factors of an integer n are less than a positive real number B, then n is called B-smooth. The Pohlig–Hellman algorithm solves the DLP efficiently when n is a smooth number.

Let $n = \prod_{q\in P} q^{e_q}$ be the factorization of n for a set P of primes. The main idea of the Pohlig–Hellman algorithm is to calculate $\alpha \pmod{q^{e_q}}$ for each $q \in P$ for $\alpha = \log_g h$. Then, $\alpha \in \mathbb{Z}_n$ can be easily recovered using the CRT.

Assume g and $h = g^\alpha$ are given. Considering a prime divisor $q \in P$, there exist $c_0, c_1, \ldots, c_{e_q-1} \in [0, q)$ satisfying $\alpha \equiv c_0 + c_1 q + \cdots + c_{e_q-1}q^{e_q-1} \pmod{q^{e_q}}$. The coefficients $c_0, c_1, \ldots, c_{e_q-1}$ are determined inductively as follows. First, from the equations $\alpha \equiv c_0 \pmod q$ and $\left(g^{\frac{p-1}{q}}\right)^{c_0} = h^{\frac{p-1}{q}}$, compute c_0 in $O\left(\sqrt{q}\right)$ using the BSGS algorithm. Note that two elements $g^{\frac{p-1}{q}}$ and $h^{\frac{p-1}{q}}$ are contained in $H = \langle g^{\frac{p-1}{q}}\rangle$, which is a subgroup of G of prime order q. Therefore, $c_0 \in [0, q)$ is uniquely determined. After calculating $c_0, c_1, \ldots, c_{i-1}$, the next coefficient c_i is obtained from the equations $\alpha \equiv c_0 + c_1 q + \cdots + c_i q^i \pmod{q^{i+1}}$ and $g^{(c_0+c_1q+\cdots+c_iq^i)\frac{p-1}{q^{i+1}}} = h^{\frac{p-1}{q^{i+1}}}$, which is equivalent to $\left(g^{\frac{p-1}{q}}\right)^{c_i} = g^{-(c_0+c_1q+\cdots+c_{i-1}q^{i-1})\frac{p-1}{q^{i+1}}}h^{\frac{p-1}{q^{i+1}}}$. This requires $O\left(\sqrt{q}\right)$ exponentiations using the BSGS algorithm. Repeating this process for all

$q \in P$, every modulus $\alpha \pmod{q^{e_q}}$ is obtained in $O\left(\sum_{q\in P} e_q\sqrt{q}\right)$ exponentiations, and $\alpha \in \mathbb{Z}_n$ can then be recovered.

Pollard's rho algorithm

The BSGS algorithm requires $O(\sqrt{n})$ memory. Pollard's rho algorithm [31] effectively reduces the necessary storage size to $O(1)$.

For a given g and $h = g^\alpha$, Pollard's rho algorithm uses a function $f : G \to G$, where G is partitioned into three sets S_0, S_1, S_2 of approximately equal size. The function f is constructed in such a way that the exponents of g and h are traceable, i.e., it is easy to compute $(x_{i+1}, \beta_{i+1}, \gamma_{i+1})$ from (x_i, β_i, γ_i) for $x_{i+1} := f(x_i)$ and $x_i = g^{\beta_i}h^{\gamma_i}$. A typical example of $f(x)$ is as follows:

$$x_{i+1} := f(x_i) = \begin{cases} hx_i, & x_i \in S_0 \\ x_i^2, & x_i \in S_1 \\ gx_i, & x_i \in S_2 \end{cases}$$

In this case, the exponents β_i and γ_i are traceable in the following ways:

$$\beta_{i+1} = \begin{cases} \beta_i, & x_i \in S_0 \\ 2\beta_i, & x_i \in S_1 \\ \beta_i + 1, & x_i \in S_2 \end{cases} \quad \text{and} \quad \gamma_{i+1} = \begin{cases} \gamma_i + 1, & x_i \in S_0 \\ 2\gamma_i, & x_i \in S_1 \\ \gamma_i, & x_i \in S_2 \end{cases}.$$

Since G is a finite set, the sequence $\{x_1, x_2, \dots\}$ obtained by evaluating f iteratively must contain a cycle. To find a collision, we compute the pair (x_i, x_{2i}) from (x_{i-1}, x_{2i-2}), and check if $x_i = x_{2i}$. We repeat this process iteratively until a collision $x_i = x_{2i}$ is found. This method is called Floyd's cycle-finding algorithm, and it requires $O(\sqrt{n})$ computation time.

If a collision $x_i = x_{2i}$ occurs for some $i > 0$, then the DLP is solved from $\alpha = -(\beta_i - \beta_{2i})(\gamma_i - \gamma_{2i})^{-1}$, unless $\gcd(\gamma_i - \gamma_{2i}, n) > 1$. Under the assumption that f behaves as a random function, Pollard's rho algorithm solves the DLP in $O(\sqrt{n})$ group operations with negligible storage.

The r-adding walk method is a generalized version of Pollard's rho algorithm that uses a function with G partitioned into r disjoint sets. Experiments have shown that the 20-adding walk is very close to the random walk [38].

One way of speeding up the collision detection is to use distinguished points [32]. A distinguished point is an element of G satisfying a specific condition that is easy to detect. During the algorithm, we check and store only the distinguished points (x_i, β_i, γ_i). If Θ denotes the proportion of distinguished points, then the total complexity is increased by Θ^{-1} steps, and the expected number of comparisons is lowered by a factor of Θ. The size of the set of distinguished points can be managed by setting an appropriate condition.

Pollard's kangaroo algorithm

Pollard's kangaroo algorithm solves the DLP when the discrete logarithm $\alpha \in [0, n)$ is contained in a certain interval $[a, b]$. The choice $a = 0, b = n - 1$ for all α is possible, but Pollard's rho algorithm is more efficient in this case.

In Pollard's kangaroo algorithm, we first precompute $g^{e_i}, 1 \le i \le r$ for some small integers $e_1, \dots, e_r$, whose sizes are approximately $\sqrt{b-a}$. Let $f : G \to \{1, 2, \dots, r\}$ be a pseudorandom function. For a suitable integer N, compute x_N as follows:

$$x_0 = g^a,\ x_{i+1} = x_i g^{e_{f(x_i)}} \ \text{ for } \ i = 0, 1, \dots, N-1.$$

Then, until a collision $y_j = x_N$ is detected, compute the following:

$$y_0 = h,\ y_{j+1} = y_j g^{e_{f(y_j)}} \ \text{ for } \ j = 0, 1, \dots, N-1.$$

The sequence $\{x_0, x_1, \dots\}$ is called a tame kangaroo, and $\{y_0, y_1, \dots\}$ is a wild kangaroo. Since the mean step size is $m = (\sum_{i=1}^{r} e_i)/r \approx \sqrt{b-a}$, the wild kangaroo meets the tame kangaroo with probability $1/m$. The complexity of the algorithm is $O(\sqrt{b-a})$.

2.2 Non-generic algorithms

In this subsection, we discuss non-generic algorithms for solving the DLP. These can generally be used only in specific groups such as $\mathbb{Z}_p^*$ or $\mathbb{F}_q^*$ for prime power q. Although these algorithms have a restricted group structure, they are more efficient than the generic algorithms.

The index calculus algorithm is an efficient way to solve the DLP when $G = \mathbb{Z}_p^*$. It consists of two steps: sieving and descent. In the sieving phase, we precompute the discrete logarithms of the small primes by finding sufficiently many relations of small primes. In the descent phase, the discrete logarithm of an arbitrary element is calculated. This algorithm runs in subexponential time. The idea of the original index calculus algorithm has been improved to the number field sieve and function field sieve algorithms [1, 19, 20, 23], which are optimized to solve the DLP very efficiently over an appropriately sized field $\mathbb{F}_{p^k}$. In particular, the complexity is very low when the characteristic p is small.

Index calculus algorithm

Consider the index calculus algorithm over a multiplicative group $G = \mathbb{Z}_p^*$. The index calculus algorithm is a probabilistic algorithm based on the prime factorization of integers. Suppose that g is a fixed generator of G. Take a suitable bound B, and let $q_0 = -1$ and $q_1 = 2 < q_2 = 3 < \cdots < q_d$ be the primes less than B. The index calculus algorithm first precomputes the DLP $g^{\beta_i} = q_i$ for $1 \le i \le d$ as follows. For a randomly chosen $\beta \in \mathbb{Z}_{p-1}$, compute the factorization of g^β modulo p. If

$g^\beta = \prod_{i=0}^{d} q_i^{e_i}$ is a B-smooth number, we have the equation $\beta = e_0\beta_0 + \cdots + e_d\beta_d$ in $\mathbb{Z}_{p-1}$; otherwise, try another value of β. Repeating this process many times, we obtain $d+1$ linearly independent equations. Then, the discrete logarithms of q_i are recovered from the linear algebra.

Now, for a given $h = g^\alpha$, we repeatedly choose random elements $\gamma \in \mathbb{Z}_{p-1}$ until $hg^\gamma \pmod p$ can be expressed as a product of primes less than B. If we find such a γ, then α is determined by $hg^\gamma = \prod_{i=0}^{d} q_i^{f_i}$ and $\alpha = -\gamma + \sum_{i=0}^{d} f_i\beta_i$. To compute the asymptotic complexity of the index calculus algorithm, we use the distribution of smooth integers. The logarithm of the probability that a random integer less than p is B-smooth is approximately $-u \log u$ for $u = \frac{\log p}{\log B}$. The complexity equals $L_p[1/2, \sqrt{2}]$ for the optimal bound $B = L_p[1/2]$. Here, L-notation is defined as

$$L_Q[\theta, c] = \exp\left[(c + o(1))(\log Q)^\theta (\log\log Q)^{1-\theta}\right]$$

for $c > 0$ and $0 \le \theta \le 1$. Note that L_Q is a polynomial function of $\log Q$ when $\theta = 0$, and an exponential function of $\log Q$ when $\theta = 1$. The asymptotic complexity $L_p[1/2, \sqrt{2}]$ of the index calculus algorithm is a subexponential function of $\log p$.

Number field sieve and function field sieve

In the index calculus algorithm, the factor bases are small primes, and the individual logarithm is found by randomizing a given integer to a smooth integer. There are many variants of the index calculus algorithm, such as the number field sieve (NFS) and the function field sieve (FFS) for solving the DLP over $\mathbb{F}_p^*$ or $\mathbb{F}_q^*$ for $q = p^n$. In addition, some variants have been optimized to binary fields.

The following is a simple variant of the index calculus algorithm. Let $q = p^n$ for some prime p and a positive integer n. Assume that p is small compared to q. The multiplicative group $G = \mathbb{F}_q^*$ of the finite field $\mathbb{F}_q$ is cyclic, and it is used in many cryptographic schemes for a base group of the DLP. A typical way to represent the finite field is $\mathbb{F}_q = \mathbb{F}_p[t]/(f)$ for a monic irreducible polynomial $f \in \mathbb{F}_p[t]$ with $\deg f = n$. Assume that $g \pmod f \in \mathbb{F}_q$ is a generator of $\mathbb{F}_q^*$. In the sieving phase, we precompute the discrete logarithms of "small" factors, and use them to calculate the discrete logarithm of an arbitrary element. In the polynomial ring $\mathbb{F}_p[t]$, a polynomial is considered to be *small* if it has a low degree. Set a suitable bound B, and let $P = \{f_1, f_2 \ldots, f_d\} \subset \mathbb{F}_p[t]$ be the set of all monic irreducible polynomials with degrees that are less than B. For randomly chosen $\beta \in \mathbb{Z}_{q-1}$, compute $g^\beta \pmod f \in \mathbb{F}_q$, and try to factorize this into elements of P to find a relation with the $\log_g f_i$'s. Repeat this process until all $\log_g f_i$'s have been calculated. In the descent phase, the individual logarithm of an arbitrary element $h \pmod f \in \mathbb{F}_q$ is obtained when an element $\gamma \in \mathbb{Z}_{q-1}$ is found such that $hg^\gamma \pmod f$ can be expressed as a product of elements of P.

The FFS is similar to the above algorithm, but it also uses some additional techniques. Since the finite field $\mathbb{F}_q$ can be represented in many ways, we can increase

the probability of finding a smooth element and a relation. Therefore, the relation collection step takes less time, and the asymptotic complexity of the FFS becomes $L_q[1/3, (32/9)^{1/3}]$ when $p \leq L_q[1/3]$. In the case of $p > L_q[1/3]$, the best known algorithm is the NFS. Its asymptotic complexity is also $L_q[1/3]$, but the coefficient c is larger. Compared to the index calculus algorithm, the coefficient is lowered to $\theta = 1/3$.

There are many additional techniques that make the sieving and descent phases more efficient. For example, in the sieving phase, we can choose two generators x, y of $\mathbb{F}_q$ with a simple relation, and set the factor base as the set of low-degree polynomials of x or y. In the descent phase, an arbitrary element h may be transformed to an element of the form z_1/z_2, such that the z_i's are products of polynomials of lower degrees. The total complexity of the algorithm depends on the sizes of p and $q = p^n$: when p is larger, the sieving phase is harder, but the descent phase is easier.

In most cases, the complexity is $L_q[1/3, c]$ for some constant $c > 0$, but optimized algorithms can achieve lower complexities under specific conditions. The best-known algorithm has quasi-polynomial complexity [1] when the base field has a small characteristic.

Since this line of research is still active, readers are advised to refer to recent results for the current status of this field.

3 The DLPwAI and Cheon's algorithm

As seen in the previous section, the hardness of the original DLP is well understood, and so cryptosystems based on this problem with appropriate parameters are believed to be secure. In recent decades, many cryptosystems based on variants of the DLP have been proposed. These variants can weaken the security of the original DLP, but they are widely used because of their flexibility, which enhances the functionality of the cryptosystems. Many of these variants can be reduced to the DLPwAI, which seeks to find $\alpha \in \mathbb{Z}_p$ for given $g, g^\alpha, \ldots, g^{\alpha^d}$. In the generic group model [4, 35], the lower bound of the complexity of solving this problem is $\Omega(\sqrt{p/d})$, which is less than $\Omega(\sqrt{p})$, the generic lower bound for the original DLP. There are some generic algorithms for the original DLP that achieve this minimum complexity, whereas none of the known algorithms solve the DLPwAI in $O(\sqrt{p/d})$ for arbitrary d and p. The first attack on the DLPwAI was introduced by Brown and Gallant [9] and Cheon [11, 12] independently. They proposed an algorithm that achieves the lower bound for the DLPwAI in certain cases. This article follows the method of Cheon's algorithm. Cheon's algorithm was later generalized by Satoh [34] and Kim et al. [24].

3.1 $p - 1$ cases

Assume that three elements $g, g_1 = g^\alpha$, and $g_d = g^{\alpha^d}$ are given for a divisor d of $p-1$. The main idea of Cheon's algorithm is to exploit the fact that α^d is contained in the

subgroup of $\mathbb{Z}_p^*$ of small order $\frac{p-1}{d}$. By applying the BSGS algorithm on this smaller group, we can recover α^d. Then, α is recovered in a similar fashion.

To initiate Cheon's algorithm, a primitive element ξ of $\mathbb{Z}_p$ is required. Since $\mathbb{Z}_p^*$ is a cyclic group of order $p-1$, there are exactly $\phi(p-1)$ primitive elements in $\mathbb{Z}_p$. A randomly chosen element in $\mathbb{Z}_p^*$ has a probability $\frac{\phi(p-1)}{p-1} \geq \frac{1}{6\log\log(p-1)}$ of being a primitive element, which is sufficiently large. Thus, it may be assumed that a primitive element ξ of $\mathbb{Z}_p$ can be found efficiently if the factorization of $p-1$ is known.

Theorem 3.1 ([12]). *Let d be a divisor of $p-1$. For given $g, g_1 = g^\alpha$, and $g_d = g^{\alpha^d}$, we can solve α deterministically in $O\left(\sqrt{\frac{p-1}{d}} + \sqrt{d}\right)$ exponentiations with storage $O\left(\max\{\sqrt{\frac{p-1}{d}}, \sqrt{d}\}\right)$.*

Proof. Consider a primitive element ξ of $\mathbb{Z}_p$. Define $\zeta = \xi^d$ and $m = \lceil\sqrt{\frac{p-1}{d}}\rceil$. There exist two integers $k_1 \in [0, d)$ and $k_2 \in [0, \frac{p-1}{d})$ such that $\alpha = \xi^{\frac{p-1}{d}k_1 + k_2}$. We will calculate k_1 and k_2 using two independent BSGS algorithms.

First, we find k_2 using the BSGS algorithm. From $\alpha^d = \xi^{dk_2} = \zeta^{k_2}$ and $g_d = g^{\alpha^d} = g^{\zeta^{k_2}}$, there exist two integers $0 \leq u_2, v_2 \leq \lfloor\sqrt{\frac{p-1}{d}}\rfloor$ such that $k_2 = mu_2 + v_2$, or equivalently $\alpha^d\zeta^{-v_2} = \zeta^{mu_2}$ and $g_d^{\zeta^{-v_2}} = g^{\zeta^{mu_2}}$. Two integers u_2 and v_2 are determined in $O\left(\sqrt{\frac{p-1}{d}}\right)$ exponentiations. After finding k_2, we again use the BSGS algorithm, and determine k_1 in $O(\sqrt{d})$ exponentiations from the equation $g_1 = g^\alpha = g^{\xi^{\frac{p-1}{d}k_1 + k_2}}$. The total complexity is $O\left(\sqrt{\frac{p-1}{d}} + \sqrt{d}\right)$ exponentiations, with $O\left(\max\{\sqrt{\frac{p-1}{d}}, \sqrt{d}\}\right)$ storage required for elements of G. □

Note that the total complexity $O\left(\max\{\sqrt{\frac{p-1}{d}}, \sqrt{d}\}\right)$ of Cheon's $p-1$ algorithm can be lowered to $O\left(p^{1/4}\right)$ when $d \approx \sqrt{p}$.

Based on Pollard's kangaroo algorithm, Cheon also proposed a probabilistic algorithm needing less storage [12]. Applying this idea to the $p-1$ algorithm, the complexity becomes $O\left(\sqrt{\frac{p-1}{d}} + \sqrt{d} + \Theta^{-1}\right)$ with storage of $O\left(\Theta \cdot \max\{\sqrt{\frac{p-1}{d}}, \sqrt{d}\}\right)$, where Θ denotes the proportion of distinguished points of Pollard's kangaroo algorithm.

The exponentiations in the total complexity of Theorem 3.1 can be converted to multiplications [26]. Refer to [33] for an implementation of Cheon's algorithm.

In [12], Cheon found one more case in which the DLPwAI can be solved efficiently by a similar algorithm. The previous algorithm works because the group $\mathbb{F}_p^*$ has a small subgroup of order $\frac{p-1}{d}$ when d is a divisor of $p-1$. By slightly modifying this idea, it is possible to find α from $g, g^\alpha, \ldots, g^{\alpha^{2d}}$ when d is a divisor of $p+1$ using a quadratic extension of $\mathbb{F}_p$. The details are omitted here, because it is more natural to consider $p+1$ as a specific case of the generalized algorithms in the next section.

3.2 Generalized algorithms

The idea of Cheon's algorithm is to embed an element in $\mathbb{F}_p$ to an element of an extension field of $\mathbb{F}_p$. More precisely, the discrete logarithm $\alpha \in \mathbb{F}_p$ is embedded into an element in $\mathbb{F}_p$ for the case $\Phi_1(p) = p-1$. Cheon's algorithm is efficient when $p-1$ has a small divisor d with given parameters $g, g^\alpha, \ldots, g^{\alpha^d}$.

Satoh [34] extended Cheon's algorithm to cases of $\Phi_k(p)$ for $k \geq 3$ using the embedding of $\mathbb{F}_p$ into $GL(k, \mathbb{F}_p)$. Recently, Kim et al. [24] realized that Satoh's embedding is essentially the same as the embedding of $\mathbb{F}_p$ into $\mathbb{F}_{p^k}$, and showed that, in most cases, this generalization cannot be faster than the square-root complexity algorithms, such as Pollard's rho algorithm, when $k \geq 3$.

Satoh's generalization

The main idea of Cheon's $p+1$ algorithm is to construct an embedding of $\mathbb{F}_p$ into its quadratic extension $\mathbb{F}_p[\theta]$. Satoh tried to generalize Cheon's algorithm using an embedding of $\mathbb{F}_p$ into a general linear group $GL(k, \mathbb{F}_p)$. This algorithm is introduced only briefly here, since it is simplified and analyzed by Kim et al. in the next section.

Definition 3.2. For a given positive integer ν, we define the p-norm $\|\nu\|_p$ as the sum of ν_i's, where each ν_i is an integer satisfying $0 \leq \nu_i < p$ and $\nu = \sum_{i \geq 0} \nu_i p^i$.

For a divisor d of $\Phi_k(p)$ for some $k \geq 1$, we put $D := \Phi_k(p)/d$. Satoh's algorithm solves the DLP with inputs $g, g^\alpha, \ldots, g^{\alpha^d}$ if it is possible to find an integer u satisfying $\gcd(u, p^k-1) = 1$ and $u(p^k-1)/D \equiv \Delta - \delta \pmod{p^k-1}$, where Δ and δ are integers with small p-norms. The total complexity is given in the following theorem.

Theorem 3.3 ([34]). *Suppose that d is a divisor of $\Phi_k(p)$ for some $k \geq 1$. Moreover, assume that an integer u satisfies $\gcd(u, p^k-1) = 1$ and $u(p^k-1)/D \equiv \Delta - \delta \pmod{p^k-1}$ for some integers Δ and δ. Then, we can solve the DLPwAI in $\tilde{O}\left(k^2(k \log p + w + k^3 + \sqrt{D})\right)$ time, where $w = \|\Delta\|_p + \|\delta\|_p$.*

This theorem is rather complicated to understand. Kim et al.'s generalization in the next section covers all cases of Satoh's algorithm, but uses simpler notation. Moreover, they observed that the generalization of Cheon's algorithm is not faster than the usual DL-solving algorithm in most cases.

Kim et al.'s generalization

Let $D = \Phi_k(p)/d$ and r be an integer. Kim et al. [24] considered an embedding

$$\mathbb{F}_p \to \mathbb{F}_{p^k}, \quad \alpha \mapsto (\alpha+\theta)^{r(p^k-1)/D}$$

for an element $\theta \in \mathbb{F}_{p^k}^{\times}$ that is not in a proper subfield, and they noticed that Satoh's embedding of $\mathbb{F}_p$ into the general linear group $GL(k, \mathbb{F}_p)$ is essentially the same as the above embedding. The element $(\alpha+\theta)^{r(p^k-1)/D}$ is an element of the subgroup of $\mathbb{F}_{p^k}$ of order D, so the idea of Cheon's algorithm can be applied.

Define $E := (p^k-1)/D$, and write rE in a signed p-ary representation as $rE = \sum_i e_i p^i$, where $|e_i| < p/2$. For an integer $\nu = \sum_i \nu_i p^i$ with the signed representation, a signed sum of digits is $S_p(\nu) := \max\{S_p^+(\nu), S_p^-(\nu)\} = \max\{\sum_{\nu_i>0}\nu_i, -\sum_{\nu_i<0}\nu_i\}$.

Consider the following:

$$(\alpha+\theta)^{rE} = \frac{(\alpha+\theta)^{\sum_{e_i>0} e_i p^i}}{(\alpha+\theta)^{\sum_{e_i<0}|e_i|p^i}} = \frac{\prod_{e_i>0}(\alpha+\theta^{p^i})^{e_i}}{\prod_{e_i<0}(\alpha+\theta^{p^i})^{|e_i|}} = \frac{f_1(\alpha)\theta_1+\cdots+f_k(\alpha)\theta_k}{h_1(\alpha)\theta_1+\cdots+h_k(\alpha)\theta_k},$$

where $\{\theta_1, \ldots, \theta_k\}$ is a basis of $\mathbb{F}_{p^k}$ for $\theta_i = \theta^{i-1}$, $\deg f_i \le S_p^+(rE)$ and $\deg h_j \le S_p^-(rE)$. Since this element is in the subgroup of order D, choose a generator ζ of this group, and then apply the BSGS algorithm to find the integer $k \in [0, D)$ satisfying $(\alpha+\theta)^{rE} = \zeta^k$.

The total complexity of this algorithm is about $O\left(\sqrt{D} + S_p(rE)\right)$. Hence, to reduce the total complexity, it is necessary to find an integer r such that rE has a low signed weight. However, by [24, Theorem 4.5], this complexity is worse than the ordinary DL-solving algorithms unless all prime divisors of D are divisors of k or $p \pm 1$.

When $k = 2$, the complexity of this algorithm is meaningful. When d is a divisor of $\Phi_2(p) = p+1$, put $D = (p+1)/d$ and $E = (p-1)d$. The signed weight of $E = dp - d$ is equal to d, which is sufficiently small. This corresponds to the case $r = 1$ of the above algorithm. Therefore, we can solve the DLPwAI in $O\left(\sqrt{\frac{p+1}{d}} + d\right)$ exponentiations with storage $O\left(\max\{\frac{p+1}{d}, \sqrt{d}\}\right)$ when d is a divisor of $p+1$, and $g, g^{\alpha}, \ldots, g^{\alpha^d}$ are given. Note that the total complexity $O\left(\sqrt{\frac{p+1}{d}} + d\right)$ can be lowered to $O(p^{1/3})$ when $d \approx p^{1/3}$.

4 Polynomials with small value sets

In this section, we introduce an approach to solve the DLPwAI without using the embedding technique. The idea, first proposed by Cheon and Kim [13, 14], is to consider a function mapping by a polynomial over $\mathbb{F}_p$.

We briefly describe the idea: first, we compute two lists $\{g^{f(r_1\alpha)}, \ldots, g^{f(r_m\alpha)}\}$ and $\{g^{f(s_1)}, \ldots, g^{f(s_m)}\}$ for given $g, g^{\alpha}, \ldots, g^{\alpha^d}$ and random $r_i, s_j \in \mathbb{F}_p$. If there exists a collision between the two lists, say $g^{f(r_i\alpha)} = g^{f(s_j)}$, then we solve the equation $f(r_i\alpha) = f(s_j)$ for the indeterminate α. Since the degree of $f(x)$ is d, we obtain at most d candidates for α, and so we can find a solution α in at most d searches. Hence, the important parts of the algorithm are: (1) to obtain a polynomial such that the expected number of m until a collision occurs is small; and (2) to compute the list $g^{f(r_1\alpha)}, \ldots, g^{f(r_m\alpha)}$ efficiently for the given $g, g^{\alpha}, \ldots, g^{\alpha^d}$.

This algorithm also gives a solution when $p \pm 1$ has an appropriate divisor d, as in Cheon's algorithm.

4.1 Fast multipoint evaluation in a blackbox manner

Let $f(x)$ be a polynomial over a field $\mathbb{F}$ of degree d. Then, we can compute $f(r_1), \ldots, f(r_m)$ in $O(\max\{m, d\} \log^2 d)$ field operations using the fast mulitipoint evaluation method. The fast multipoint evaluation method involves fast multiplication methods, fast Fourier transformations, fast polynomial divisions, and so on. For more details on this, refer to [40]. Even though the coefficients of $f(x)$ are given in exponentiated form, we can compute the multipoint evaluation efficiently. A precise description is as follows.

Proposition 4.1. *If $g^{a_0}, \ldots, g^{a_d}$ are given for a polynomial $f(x) = a_d x^d + \cdots + a_1 x + a_0$, then we can compute $g^{f(r_1)}, \ldots, g^{f(r_m)}$ in $O(m \log^2 d)$ group exponentiations for $m \geq d$.*

The proof is easily obtained from the proof of the fast multipoint evaluation method by replacing the field multiplications/additions with the group exponentiations/multiplications in the proof. The following is a direct consequence of Proposition 4.1.

Corollary 4.2. *For given $g, g^{\alpha}, \ldots, g^{\alpha^d}$ and a polynomial $f(x) \in \mathbb{F}_p[x]$ of degree d, we can compute $g^{f(r_1\alpha)}, \ldots, g^{f(r_m\alpha)}$ in $O(d + m \log^2 d)$ group exponentiations for any elements $r_1, \ldots, r_m$ in $\mathbb{F}_p$.*

Proof. We can obtain $g^{a_0}, (g^{\alpha})^{a_1}, \ldots, (g^{\alpha^d})^{a_d}$ in d exponentiations from $g, g^{\alpha}, \ldots, g^{\alpha^d}$ and $f(x)$. Now, let $h(x) := f(x\alpha) = a_d\alpha^d x^d + \cdots + a_1\alpha x + a_0$ and apply Proposition 4.1 to $h(x)$. □

4.2 An approach using polynomials of small value sets

In this section, we discuss the reduction of the DLPwAI into finding a polynomial with a small value set.

Theorem 4.3 ([14]). *Let $g, g^{\alpha}, \ldots, g^{\alpha^d}$ be given, and let $f(x) := f_0 + f_1 x + \cdots + f_d x^d$ be a polynomial over $\mathbb{F}_p$ of degree d. Define the value set of $f(x)$ by $V(f) :=$*

$\{f(x) : x \in \mathbb{F}_p\} = \{a_1, \ldots, a_t\}$, *where* t *is the size of the value set. If the sizes of the preimages* $|f^{-1}(a_i)|$ *are almost uniform for* $1 \leq i \leq t$, *then we can solve* α *in* $O(m \log^2 d + d \log p)$ *group operations, where* $m := \Theta(\sqrt{t})$.

Proof. First, we compute two lists $\{g^{f(r_i\alpha)} : i = 1, 2, \ldots, m\}$ and $\{g^{f(s_j)} : j = 1, 2, \ldots, m\}$ for randomly chosen r_i and s_j in time complexity $O(d + m \log^2 d)$ using the proposition in the previous section. Since the preimage of each a_i has almost the same size, $f(r_i\alpha)$ and $f(s_j)$ behave as random elements chosen from $V(f)$, and the two lists have a high probability of collision by the birthday paradox.

After a collision has occurred, solving the equation $f(r_i\alpha) = f(s_j)$ with respect to the indeterminate α gives the desired solution. We can find roots in $\mathbb{F}_p$ of a polynomial of degree d over $\mathbb{F}_p$ using an expected number of $\tilde{O}(d \log p)$ operations. Since there exist at most d solutions to this equation, we can find α in $O(d)$ group exponentiations, and compare these values with g^α. □

By virtue of Theorem 4.3, solving the DLPwAI is reduced to finding a polynomial with a small value set. Finding such polynomials is a well-known research topic in the field of number theory.

Consider the simple example $f(x) = x^d$, where $d|(p-1)$. Then, the size of $V(f)$ is $\frac{p-1}{d} + 1$, and by applying the above theorem, we solve the DLPwAI in $\tilde{O}\left(\sqrt{\frac{p-1}{d}}\right)$. Note that any polynomial of form $(x+b)^d + c$ for $b, c \in \mathbb{F}_p$ has a value set of the same size when $d|(p-1)$ [10]. Any polynomial of degree d has a value set of size at least $\lfloor \frac{p-1}{d} \rfloor + 1$. The above polynomials attain the lower bound, and they are the only polynomials that attain this minimum [10].

However, it is not so easy to find a polynomial with a small value set: it is known that the expected size of the value set over the polynomials of degree d is about $\frac{p}{e}$, where e is the Euler constant [39]. In [21], polynomials with a value set of size less than $2p/d$ were classified for $d < p^{1/4}$. A good reference on topics related to polynomials with small value sets is [36].

Fortunately, although the value set of a given polynomial is not small, it is possible to extend the idea in some cases. Consider the subset $S \subset \mathbb{F}_p$ and $V_S(f) := \{f(x) : x \in S\}$. Applying Theorem 4.3 for the set $V_S(f)$ instead of $V(f)$, the same arguments as in the proof hold when elements $r_i\alpha$ and s_j are regarded as random elements over S. Since the random element $x \in \mathbb{F}_p$ is in S with probability $|S|/p$, the total complexity becomes $O\left(\frac{p}{|S|} \cdot m\right)$, where $m = \sqrt{|V_S(f)|}$. For example, the Dickson polynomial of degree d is a d-to-1 function on the set of size $p/2$ when d divides $p+1$ [16]. Thus, we can compute the DLPwAI in $O(2 \cdot \sqrt{\frac{p}{2d}})$ group operations.

More generally, we can reduce the DLPwAI to find a polynomial whose substitution polynomial $f(x) - f(y) = 0$ has many absolutely irreducible factors [25].

5 Approach using the rational polynomials: Embedding to elliptic curves

In this section, we describe a method proposed in [25] that uses an embedding from the finite field into an elliptic curve group.

We assume that $E(\mathbb{F}_p)$ is cyclic in this section, and its order has a proper divisor d (the size of d will be determined later). Let H be the subgroup of order δ in $E(\mathbb{F}_p)$, where $|E(\mathbb{F}_p)| = d \cdot \delta$.

Let $\alpha \in \mathbb{F}_p$ be the discrete logarithm to be solved. There exists a point P in $E(\mathbb{F}_p)$ whose x-coordinate is $(r\alpha)$ for some $r \in \mathbb{F}_p$. Then, the x-coordinate of dP can be computed using the d-th division polynomials $\phi_d(x)$ and $\psi_d^2(x)$ [37] with only the x-coordinate of the base point:

$$x[dP] = \frac{\phi_d(r\alpha)}{\psi_d^2(r\alpha)},$$

where $x[P]$ denotes the x-coordinate of $P \in E(\mathbb{F}_p)$. Since dP lies in a subgroup H, we can find α by choosing $\widetilde{P}$ from H such that $\widetilde{P} = dP$, and solving the equation $\frac{\phi_d(r\alpha)}{\psi_d^2(r\alpha)} = x[\widetilde{P}]$.

To find a proper $\widetilde{P}$, we proceed as follows: first, choose random points $\widetilde{P}_1, \widetilde{P}_2, \ldots, \widetilde{P}_m$ uniformly from H, and compute a list

$$L_1 := \{g^{x[\widetilde{P}_i]} : 1 \le i \le m, \widetilde{P}_i \in H\}.$$

Independently, choose random $r_1, r_2, \ldots, r_m$ uniformly from $\mathbb{F}_p$, and compute a list

$$L_2 := \{g^{\phi_d(r_j\alpha)/\psi_d^2(r_j\alpha)} : 1 \le j \le m, r_j \in \mathbb{F}_p\}.$$

Since the order of H is δ, we can expect to find a collision between the lists in $m = O(\sqrt{\delta})$ by the birthday paradox.

However, the main obstacle in this algorithm is that computing list L_2 seems hard, although we can compute each $g^{\phi_d(r_j\alpha)}$ and $g^{\psi_d^2(r_j\alpha)}$ with $g, g^{\alpha}, \ldots, g^{\alpha^{d^2}}$ using Proposition 4.1.

6 Generalized DLPwAI

The main topic of this section is a generalization of the DLPwAI, called the generalized DLPwAI (GDLPwAI). The GDLPwAI aims to determine α for given $g^{\alpha^{e_1}}, \ldots, g^{\alpha^{e_d}}$, where $e_1, \ldots, e_d$ are arbitrary integers. The DLPwAI can be considered as the special case of the GDLPwAI with $e_i = i$.

In Section 4, a polynomial f of degree $d \approx p^{1/3}$ with a small value set was needed to solve the DLPwAI. Unfortunately, such polynomials are extremely rare.

Cheon, Kim, and Song [15] noticed that it is easy to find a polynomial with a small value set when the degree is not so small. Unlike in the previous section, fast multipoint evaluation cannot be applied. Instead, they chose a polynomial $f(x)$ so that there exist elements ζ_i satisfying $f(\zeta_i x) = \zeta_i f(x)$. Then, it is possible to evaluate many points by a simple operation to multiply ζ_i. Moreover, they derived a heuristic algorithm to solve the GDLPwAI efficiently when $K = \{e_1, \ldots, e_d\}$ is a multiplicative subgroup of $\mathbb{Z}_{p-1}^{\times}$ using a K-group action on $\mathbb{Z}_p^*$. In this section, we assume K is a multiplicative subgroup of $\mathbb{Z}_{p-1}^{\times}$.

6.1 Representation of a multiplicative subgroup of $\mathbb{Z}_{p-1}^{\times}$

Before stating our theorem, we define a representation λ for a multiplicative subgroup K of $\mathbb{Z}_{p-1}^{\times}$. We may regard K as part of some arithmetic sequences starting from 1, and want to take λ to be the largest of the intervals of such sequences.

Let $\pi : \mathbb{Z} \to \mathbb{Z}_{p-1}$ be the canonical projection. For a multiplicative subgroup K of $\mathbb{Z}_{p-1}^{\times}$, let I_K be the ideal of $\mathbb{Z}_{p-1}$ generated by $\{k - 1 \in \mathbb{Z}_{p-1} : k \in K\}$. Then, $\pi^{-1}(I_K)$ is an ideal of $\mathbb{Z}$, and we define λ as a unique nonnegative generator of this ideal. We often simply write $\lambda = \gcd(K - 1)$, since λ is the greatest common divisor of $K - 1 := \{k - 1 \in \mathbb{Z} : k \in K\}$, where each element of $K - 1$ is considered as an integer in the set $\{1, 2, \ldots, p - 1\}$.

Note that λ is an even divisor of $p - 1$, since $p - 1 \in K - 1$, and elements of S are even. Moreover, every element of K is of the form $1 + \lambda m$ for some $m \in \mathbb{Z}_{p-1}$. We also define $K_\lambda = (1 + \lambda\mathbb{Z}_{p-1}) \cap \mathbb{Z}_{p-1}^{\times}$ to be the set of unit elements k of $\mathbb{Z}_{p-1}$ such that $k - 1$ is a multiple of λ. It is easy to check that K_λ is closed under multiplication and the inverse function, and is therefore a multiplicative subgroup of $\mathbb{Z}_{p-1}^{\times}$. Note that K is a subgroup of K_λ, and the index $\kappa = [K_\lambda : K]$ is a positive integer.

6.2 A group action on $\mathbb{Z}_p^*$ and polynomial construction

Define a group action $K \times \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ by $(k, x) \mapsto x^k$. The orbit generated by x is the set $\{x^k : k \in K\}$, denoted by x^K. For a primitive element ξ of $\mathbb{Z}_p$, and $\zeta = \xi^{\frac{p-1}{\lambda}}$, the set of fixed points of this group action is

$$(\mathbb{Z}_p^*)_K = \{x \in \mathbb{Z}_p^* : x^k = x \text{ for all } k \in K\} = \{x \in \mathbb{Z}_p^* : x^\lambda = 1\} = \langle\zeta\rangle,$$

where $\lambda = \gcd(K - 1)$ is the representation of K defined above, and $\langle\zeta\rangle$ denotes the cyclic subgroup generated by ζ. Now, we define a polynomial $f = f_K$ on $\mathbb{Z}_p$ by $f(x) = \sum_{k \in K} x^k$. If x and y are contained in the same orbit, then $y = x^k$ for some $k \in K$ and $f(y) = f(x)$. Moreover, $f(\zeta^i x) = \zeta^i f(x)$ is satisfied for all $0 \le i < \lambda$. Therefore, f has the same value in the same orbit, and we can calculate the value of $f(\zeta^i x)$ efficiently from $f(x)$. These properties will be used in the following theorem.

6.3 Main result

Combining the results above, we obtain the following theorem. It is assumed that the computational cost for the multiplications in G is a constant times the cost of the multiplications in $\mathbb{Z}_p$.

Theorem 6.1. *Let K be a multiplicative subgroup of $\mathbb{Z}_{p-1}^{\times}$, $\lambda = \gcd(K-1)$ be its representation, and $\kappa = [K_\lambda : K]$ be the index defined above. Then, we can solve for α in $O\left(\frac{p}{\lambda|K|}(\sqrt{\lambda}+|K|)\right)$ exponentiations in $\mathbb{Z}_p$ for given $\left\{\left(k, g^{\alpha^k}\right) : k \in K\right\}$ if $|\alpha^K| = |K|$ and $f(\alpha) \neq 0$.*

Proof. First, calculate $g^{f(\alpha)} = \prod_{k\in K} g^{\alpha^k}$. Three elements $\beta \in \mathbb{Z}_p^*$, $t \in [0,\lambda)$, and $k \in K$ satisfying $\alpha^k = \zeta^t\beta$ will be determined in the next step. Take a random β from $\mathbb{Z}_p^*$, and try to find $t \in [0,\lambda)$ satisfying $\alpha^K = \zeta^t\beta^K$. Since $|\alpha^K| = |K|$, there are $\lambda|K|$ distinct elements of the form $\zeta^i\alpha^k$ in $\mathbb{Z}_p^*$, so we can find such a β with probability $\frac{\lambda|K|}{p-1}$. To find $t \in [0,\lambda)$, calculate $f(\beta)$ in $O(|K|)$ exponentiations, and use the BSGS algorithm with the relations $\alpha^K = \zeta^t\beta^K$ and $g^{f(\alpha)} = g^{\zeta^t f(\beta)}$. This can be done in $O(\sqrt{\lambda})$ exponentiations. After finding t, the exponent k can be obtained in a short time by comparing elements from the equations $\alpha^k = \zeta^t\beta$ and $g^{\alpha^k} = g^{\zeta^t\beta}$. The total complexity of this algorithm is $O\left(\frac{p}{\lambda|K|}(\sqrt{\lambda}+|K|)\right)$ exponentiations, since the expected number of repetitions is $\frac{p-1}{\lambda|K|}$. □

The complexity of the algorithm is a function of λ and $|K|$, but these are not independent parameters. The following lemma computes $|K_\lambda|$ exactly, and helps convert the total complexity to a simple form.

Lemma 6.2. *Let λ be a divisor of $p-1$. Then, $|K_\lambda| = \frac{p-1}{\lambda}\cdot\prod_{q\in Q}\left(1-\frac{1}{q}\right)$, where Q is the set of prime divisors of $p-1$ that do not divide λ.*

The product $\prod_{q\in Q}\left(1-\frac{1}{q}\right)$ is not particularly small, so it may be assumed that K_λ is a constant multiple of p/λ. From this lemma, the total complexity can be expressed as a function of λ and κ.

Corollary 6.3. *Under the same conditions as Theorem 6.1, we can solve the GDLPwAI in $O\left(\left(\kappa\sqrt{\lambda}+\frac{p}{\lambda}\right)\log p\right)$ multiplications in $\mathbb{Z}_p$.*

Note that the complexity can be lowered to $O\left(p^{1/3}\right)$ when $\lambda \approx p^{2/3}$ and κ is a constant.

Two conditions $|\alpha^K| = |K|$ and $f(\alpha) \neq 0$ are required to run the algorithm. The following lemma implies that the first condition is satisfied with a high probability.

Lemma 6.4. *If* $\gcd(\lambda, \frac{p-1}{\lambda}) = 1$*, then* $|x^K| = |K|$ *for* x *satisfying* $\mathrm{ord}_p(x^\lambda) = \frac{p-1}{\lambda}$*. Moreover, there are exactly* $\lambda\phi(\frac{p-1}{\lambda})$ *elements* x *in* $\mathbb{Z}_p^*$ *such that* $\mathrm{ord}_p(x^\lambda) = \frac{p-1}{\lambda}$*.*

The second condition is not restrictive, but it is interesting to extend the algorithm to include this case.

7 Applications and implications

7.1 Strong Diffie–Hellman Problem and Its Variants

Many cryptosystems are designed such that their security is based on variants of the Diffie–Hellman (DH) problem. These variants are seemingly weaker than the DLP or the DH problem. However, they are widely used to construct cryptosystems, as they provide more functionality. In this section, we review these variants of the DH problem. These problems are directly solved when the DLPwAI is solved.

The ℓ-Weak Diffie–Hellman Problem aims to compute $g^{1/\alpha} \in G$ for given $(g, g^\alpha, \ldots, g^{\alpha^\ell}) \in G^{\ell+1}$. The problem was introduced by Mitsunari, Sakai, and Kasahara [30].

Many other similar problems have been suggested in the bilinear setting. We consider an efficiently computable bilinear map $e : G \times G \to G_T$ for a cyclic group G_T of prime order p.

The ℓ-Strong Diffie–Hellman Problem attempts to find $(c, g^{1/(\alpha+c)})$ for any chosen $c \in \mathbb{Z}_p \backslash \{-\alpha\}$ from $(g, g^\alpha, \ldots, g^{\alpha^\ell}) \in G^{\ell+1}$. The problem was introduced by Boneh and Boyen to construct short signatures without random oracles [4], and was later used to construct short group signatures [7].

The ℓ-Bilinear Diffie–Hellman Inversion Problem focuses on finding $e(g, g)^{1/\alpha}$ for given $(g, g^\alpha, \ldots, g^{\alpha^\ell}) \in G^{\ell+1}$. Boneh and Boyen constructed a selectively secure ID-based encryption without a random oracle model, the security of which was based on this problem [3].

The ℓ-Bilinear Diffie–Hellman Exponent Problem is the problem of computing $e(g, g)^{\alpha^\ell}$ for given $(g, g^\alpha, \ldots, g^{\alpha^{\ell-1}}, g^{\alpha^{\ell+1}}, \ldots, g^{\alpha^{2\ell}}) \in G^{2\ell}$. It was first used for a hierarchical identity–based encryption with constant-size ciphertext [6], and was later employed for a public key broadcast encryption with short ciphertexts and private keys [8].

More generally, most of these problems can be translated to a setting with an asymmetric bilinear map $e : G_1 \times G_2 \to G_T$ [5].

7.2 Attack on the existing schemes using Cheon's algorithm

Blind signatures on the gap Diffie–Hellman group

A gap Diffie–Hellman (GDH) group is one in which the computational DH problem is hard, but the decisional DH problem is easy. Boldyreva proposed a blind signature scheme on the GDH group [2]. This scheme can be forged under the chosen message attack when the DLPwAI can be easily solved [12]: first, we briefly describe the scheme in [2]. Let $\alpha \in \mathbb{Z}_p$ be a secret key, and $h := g^\alpha$ be a corresponding public key. To blindly sign a message m, we randomly choose $r \in \mathbb{Z}_p$ and send $m' = H(m) \cdot g^r$ to a signer, where H is a hash function from the message space to the group G. The signer computes $\sigma' = (m')^\alpha$, and returns it to the sender. Then, the signature is obtained by computing $\sigma'/h^r = H(m)^\alpha$.

We can obtain the instance of the DLPwAI from the signature queries for the chosen message. Note that the signer has no information on the message to be signed. First, the attacker requests a signature on the message $h \cdot g^{r_1}$ for a random $r_1 \in \mathbb{Z}_p$. Then, the signer returns $(h \cdot g^{r_1})^\alpha$, from which the attacker can obtain $h^\alpha = (h \cdot g^{r_1})^\alpha / h^{r_1} = g^{\alpha^2}$. Recursively, the attacker can obtain $h^{\alpha^i} = g^{\alpha^{i+1}}$ by requesting a signature on the message $(h^{\alpha^{i-1}} \cdot g^{r_i})$ for random r_i's.

Forgery of the Boneh–Boyen signature

Consider a bilinear map $e : G \times G \to G_T$, where G and G_T are cyclic groups of prime order p, and $g \in G$ is a generator of G. Boneh and Boyen constructed a short signature scheme [4] with the bilinear map.

The public key of the Boneh–Boyen signature scheme is a pair $(g, h = g^\alpha)$ for a generator g of G, where the secret key is $\alpha \in \mathbb{Z}_p^*$. The signature corresponding to the message $m \in \mathbb{Z}_p$ is given by $\sigma = g^{1/(\alpha - m)}$. The verification follows from the relation $e(\sigma, h \cdot g^{-m}) = e(g, g)$.

Jao and Yoshida [22] noticed that the secret key α is recoverable from d signature queries using Cheon's algorithm. The idea uses the partial fraction decomposition theorem:

$$\frac{A(x)}{(x - m_1) \cdots (x - m_d)} = B(x) + \frac{e_1}{x - m_1} + \cdots \frac{e_d}{x - m_d},$$

where $m_1, \ldots, m_d \in \mathbb{Z}_p$ and $A(x)$ and $B(x)$ are polynomials in $\mathbb{Z}_p[x]$. In particular, we can compute $B(x)$ and $e_1, \ldots, e_d$ for given $m_1, \ldots, m_d$ and $A(x)$.

The attack on the Boneh–Boyen signature can be described as follows. An attacker queries d signatures on the messages $m_1, \ldots, m_d$, and obtains the corresponding signatures $\sigma_i = g^{1/(\alpha - m_i)}$ for $1 \le i \le d$.

Let $g_1 := g^{1/(\alpha - m_1) \cdots (\alpha - m_d)}$. Using the partial decomposition theorem, the attacker computes constants $e_1, \ldots, e_d$ satisfying $\frac{1}{(x-m_1)\cdots(x-m_d)} = \frac{e_1}{x-m_1} + \cdots \frac{e_d}{x-m_d}$. Thus, the attacker can compute the value of g_1 from the signatures $\sigma_i = g^{1/(\alpha - m_i)}$ and the

constants e_i. Repeatedly, the attacker obtains values of $g_1^{\alpha^i} = g^{\frac{\alpha^i}{(\alpha-m_1)\cdots(\alpha-m_d)}}$ using the partial decomposition theorem with $A(x) = x^i$.

The complexity of computing the parameters $g_1, g_1^{\alpha}, \ldots, g_1^{\alpha^d}$ (or $g_1, g_1^{\alpha}, \ldots, g_1^{\alpha^{2d}}$) is at most $O(d^2)$ exponentiations. Combining this consequence with Cheon's algorithm, the attacker can recover the secret key α in time $O\left(\sqrt{\frac{p}{d}} + d^2\right)$ when $p \pm 1$ has an appropriate divisor d.

8 Open problems and further work

In this article, several approaches to solving the DLPwAI have been described. One of the main goals of this research is to solve the problem when $\Phi_k(p)$ has an appropriate divisor d for $k \geq 3$. Currently, the best-known algorithm, Cheon's algorithm, only solves the problem when $p \pm 1$ has a small divisor d. With the approaches described in Section 4, the DLPwAI can be solved when we can find a polynomial of degree $d < p^{1/3}$ that has a value set of size $< \frac{cp}{d}$ for some fixed constant c, but the existence of such a polynomial has not yet been proved.

On the other hand, in the argument of Theorem 4.3 on the collision probability caused by the function mapping, we assumed that the distributions of the values were uniform. It is, therefore, possible to consider extending the argument to the case of non-uniform distributions. This might give rise to the problem of finding a polynomial whose substitution polynomial has lots of absolutely irreducible factors. For the non-uniform birthday problem for a constant d, we refer to [18]. However, for the DLPwAI, the argument should hold for $d < p^{1/3}$. This extension remains an open problem.

The approach described in Section 5 settles the problem whereby, given $a_i \in \mathbb{Z}_p$ for $1 \leq i \leq m$ and (g^{b_j}, g^{c_j}) for $1 \leq j \leq m$, we wish to efficiently find a collision between g^{a_i} and g^{b_j/c_j}.

If we have an algorithm for multipoint evaluation on a d-sparse polynomial (only d-coefficients are non-zero) that runs in linear time in the number of evaluated points, then the algorithm described in Section 6 can be improved. Further, any d-sparse polynomial with a small value set can be used to solve the GDLPwAI in more general instances, provided that the efficient algorithm for multipoint evaluation exists. Note that the algorithm proposed by Cheon, Kim, and Song requires these instances to have a special structure: the algorithm runs for the inputs $g^{\alpha^{e_i}}$, where each e_i is an element from a certain multiplicative group.

Independently, we may generalize the problem further: let $f_i(x)$ be any polynomial over $\mathbb{Z}_p$ for $1 \leq i \leq d$. Is there any efficient algorithm to find α for a given $g^{f_i(\alpha)}$, and what would be the lower bound for these problems in the generic group model [35]? Is it possible to reduce the DLPwAI to these generalized problems?

Currently, Cheon's $p \pm 1$ algorithm attains the lower bound in the generic group model. From the results in the current state, however, it is hard to predict the security of the problem for other cases: does an algorithm exist that can attain the lower bound

of complexity, and, if not, how far can we reduce the complexity of the problem?

Acknowledgments. The authors would like to thank Steven Galbraith for his discussion on the non-uniform birthday problem, and also extend their appreciation to Michael Zieve for discussions on polynomials with small value sets.

Bibliography

[1] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux and Emmanuel Thomé, A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic, *IACR Cryptology ePrint Archive* (2013), `http://eprint.iacr.org/2013/400`.

[2] Alexandra Boldyreva, Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme, in: *Public Key Cryptography - PKC 2003*, Lecture Notes in Computer Science 2567, pp. 31–46, Springer, 2003.

[3] Dan Boneh and Xavier Boyen, Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles, in: *Advances in Cryptology - EUROCRYPT 2004* (Christian Cachin and Jan Camenisch, eds.), Lecture Notes in Computer Science 3027, pp. 223–238, Springer, 2004.

[4] _____, Short Signatures Without Random Oracles, in: *Advances in Cryptology - EUROCRYPT 2004* (Christian Cachin and Jan Camenisch, eds.), Lecture Notes in Computer Science 3027, pp. 56–73, Springer, 2004.

[5] _____, Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups, *J. Cryptology* 21 (2008), 149–177.

[6] Dan Boneh, Xavier Boyen and Eu-Jin Goh, Hierarchical Identity Based Encryption with Constant Size Ciphertext, in: *Advances in Cryptology - EUROCRYPT 2005*, Lecture Notes in Computer Science 3494, pp. 440–456, Springer, 2005.

[7] Dan Boneh, Xavier Boyen and Hovav Shacham, Short Group Signatures, in: *Advances in Cryptology - CRYPTO 2004*, Lecture Notes in Computer Science 3152, pp. 41–55, Springer, 2004.

[8] Dan Boneh, Craig Gentry and Brent Waters, Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys, in: *Advances in Cryptology - CRYPTO 2005* (Victor Shoup, ed.), Lecture Notes in Computer Science 3621, pp. 258–275, Springer, 2005.

[9] Daniel R. L. Brown and Robert P. Gallant, The Static Diffie-Hellman Problem, *IACR Cryptology ePrint Archive* (2004), `http://eprint.iacr.org/2004/306`.

[10] L. Carlitz, D. J. Lewis, W. H. Mills and E. G. Strauss, Polynomials over Finite Fields with Minimal Value Sets, *Mathematika* 8 (1961), 121–130.

[11] Jung Hee Cheon, Security Analysis of the Strong Diffie-Hellman Problem, in: *Advances in Cryptology - EUROCRYPT 2006* (Serge Vaudenay, ed.), Lecture Notes in Computer Science 4004, pp. 1–11, Springer, 2006.

[12] _____, Discrete Logarithm Problems with Auxiliary Inputs, *J. Cryptology* 23 (2010), 457–476.

[13] ———, Discrete Logarithm in Pairing Groups, in: *Invited Talk at Pairing 2012*, 2012.

[14] Jung Hee Cheon and Taechan Kim, Discrete logarithm with auxiliary inputs, in: *MSJ-KMS Joint Meeting 2012*, 2012.

[15] Jung Hee Cheon, Taechan Kim and Yong Soo Song, A Group Action on $\mathbb{Z}_p{}^{\times}$ and the Generalized DLP with Auxiliary Inputs, in: *to appear in Selected Areas in Cryptography 2013*.

[16] W.-S. Chou., J. Gomez-Calderon and G. L. Mullen, Value Sets of Dickson Polynomials over Finite Fields, *Journal of Number Theory* 30 (1988), 334 –344.

[17] Steven D. Galbraith, *Mathematics of Public Key Cryptography*, Cambridge University Press, 2012.

[18] Steven D. Galbraith and Mark Holmes, A non-uniform birthday problem with applications to discrete logarithms, *Discrete Applied Mathematics* 160 (2012), 1547–1560.

[19] Faruk Göloglu, Robert Granger, Gary McGuire and Jens Zumbrägel, On the Function Field Sieve and the Impact of Higher Splitting Probabilities: Application to Discrete Logarithms in $\mathbf{F}_{2^{1971}}$.

[20] Faruk Göloğlu, Robert Granger, Gary McGuire and Jens Zumbrägel, Solving a 6120-bit DLP on a Desktop Computer.

[21] J. Gomez-Calderon and D. J. Madden, Polynomials with Small Value Set over Finite Fields, *Journal of Number Theory* 28 (1988), 167–188.

[22] David Jao and Kayo Yoshida, Boneh-Boyen Signatures and the Strong Diffie-Hellman Problem, in: *Pairing-Based Cryptography - Pairing 2009*, 5671, pp. 1–16, 2009.

[23] Antoine Joux, A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic, *IACR Cryptology ePrint Archive* (2013), `http://eprint.iacr.org/2013/095`.

[24] Minkyu Kim, Jung Hee Cheon and In-Sok Lee, Analysis on a generalized algorithm for the strong discrete logarithm problem with auxiliary inputs, *Mathematics of Computation* to appear.

[25] Taechan Kim and Jung Hee Cheon, A New Approach to Discrete Logarithm Problem with Auxiliary Inputs, *IACR Cryptology ePrint Archive* (2012), `http://eprint.iacr.org/2012/609`.

[26] Shunji Kozaki, Taketeru Kutsuma and Kazuto Matsuo, Remarks on Cheon's Algorithms for Pairing-Related Problems, in: *Pairing-Based Cryptography - Pairing 2007*, Lecture Notes in Computer Science 4575, pp. 302–316, Springer, 2007.

[27] Ueli M. Maurer, Towards the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Algorithms, in: *Advances in Cryptology - CRYPTO '94* (Yvo Desmedt, ed.), Lecture Notes in Computer Science 839, pp. 271–281, Springer, 1994.

[28] Ueli M. Maurer and Stefan Wolf, The Relationship Between Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms, *SIAM J. Comput.* 28 (1999), 1689–1721.

[29] Shigeo Mitsunari, Ryuichi Sakai and Masao Kasahara, A New Traitor Tracing, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E85-A (2002), 481–484.

[30] _____, A New Traitor Tracing, *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences* E85-A (2002), 481–484.

[31] J. M. Pollard, Monte Carlo Methods for Index Computation $(\mathrm{mod}\, p)$, *Mathematics of Computation* 32 (1978), pp. 918–924.

[32] Jean-Jacques Quisquater and Jean-Paul Delescaille, How Easy is Collision Search? Application to DES (Extended Summary), in: *EUROCRYPT*, pp. 429–434, 1989.

[33] Yumi Sakemi, Goichiro Hanaoka, Tetsuya Izu, Masahiko Takenaka and Masaya Yasuda, Solving a Discrete Logarithm Problem with Auxiliary Input on a 160-Bit Elliptic Curve, in: *Public Key Cryptography - PKC 2012*, Lecture Notes in Computer Science 7293, pp. 595–608, Springer, 2012.

[34] Takakazu Satoh, On Generalization of Cheon's Algorithm, *IACR Cryptology ePrint Archive* (2009), `http://eprint.iacr.org/2009/058`.

[35] Victor Shoup, Lower Bounds for Discrete Logarithms and Related Problems, in: *Advances in Cryptology - EUROCRYPT '97* (Walter Fumy, ed.), Lecture Notes in Computer Science 1233, pp. 256–266, Springer, 1997.

[36] Igor Shparlinski, *Finite Fields: Theory and Computation*, Springer, 1999.

[37] J. H. Silverman, *The arithmetic of elliptic curves*, Springer, 2009.

[38] Edlyn Teske, On Random Walks For Pollard's Rho Method, *Mathematics of Computation* 70 (2000), 809–825.

[39] Saburo Uchiyama, Note on the Mean Value of $V(f)$, *Proc. Japan Acad* 31 (1955), 199–201.

[40] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, 2003.

Author information

Jung Hee Cheon, Department of Mathematical Sciences, Seoul National University,
1 Gwanak-ro, Gwanak-gu, Seoul, Korea.
E-mail: `jhcheon@snu.ac.kr`

Taechan Kim, Department of Mathematical Sciences, Seoul National University,
1 Gwanak-ro, Gwanak-gu, Seoul, Korea.
E-mail: `yoshiki1@snu.ac.kr`

Yongsoo Song, Department of Mathematical Sciences, Seoul National University,
1 Gwanak-ro, Gwanak-gu, Seoul, Korea.
E-mail: `lucius05@snu.ac.kr`