Subset-Restricted Random Walks for Pollard rho Method on $\mathbf{F}_{p^m}$ [⋆]

Minkyu Kim, Jung Hee Cheon, and Jin Hong

ISaC and Department of Mathematical Sciences
Seoul National University, Seoul 151-747, Korea
{minkyu97,jhcheon,jinhong}@snu.ac.kr

Abstract. In this paper, we propose a variant of the Pollard rho method. We use an iterating function whose image size is much smaller than its domain and hence reaches a collision faster than the original iterating function. We also explicitly show how this general method can be applied to multiplicative subgroups of finite fields with large extension degree. The construction for finite fields uses a distinctive feature of the normal basis representation, namely, that the p-th power of an element is just the cyclic shift of its normal basis representation, when the underlying field is of characteristic p. This makes our method appropriate for hardware implementations. On multiplicative subgroups of $\mathbf{F}_{p^m}$, our method shows time complexity advantage over the original Pollard rho method by a factor of approximately $\frac{3p-3}{4p-3}\sqrt{m}$.
Through the MOV reduction, our method can be applied to pairing-based cryptosystems over binary or ternary fields. Hence our algorithm suggests that the order of subgroups, on which the pairing-based cryptosystems rely, needs to be increased by a factor of approximately m.

keywords: discrete logarithm problem, pairing, Pollard rho method, normal basis

1 Introduction

Let G be a finite cyclic group of order q generated by g. Given $h \in G$, the discrete logarithm problem (DLP) over G is to find the smallest non-negative integer x satisfying $g^x = h$. The integer x, denoted by $\log_g h$, is called the discrete logarithm of h to the base g. Given g and x, exponentiation $g^x(= h)$ can be efficiently computed through the Square-and-Multiply method, but recovering x from $h = g^x$ is considered to be difficult in many cases. Many cryptographic algorithms have been proposed under the assumption that the DLP is hard to solve under a specific presentation of the cyclic group.

The cyclic groups most widely used with cryptosystems are the multiplicative subgroups of finite fields and the subgroups of elliptic curves defined over finite fields. In practical use of finite fields, prime fields have drawn more attention, than other fields, because prime fields are easier to implement on general purpose

[⋆] A part of this paper was made public through [1].

hardware and since the index calculus attack [2] is faster on binary fields than on prime fields [6]. In contrast, with pairing-based cryptography, binary and ternary fields are widely used as the base field over which Tate or Weil pairings are defined, due to the size of their embedding degrees [3, 4, 11]. Note that the base problems on which pairing-based cryptosystems are built can be transformed into the DLP on binary or ternary fields through the MOV reduction [10, 13]. This increases the significance of the study of DLP on extension fields.

In this paper, we propose an algorithm to solve the DLP in large order extension fields. The normal basis representation is used, and this makes our algorithm suitable in hardware implementations. Our algorithm exploits the distinctive feature of the normal basis representation, namely, that the p-th power of an element is just the cyclic shift of its normal basis representation, where p is the characteristic of the underlying field. This feature was used in [8, 12, 27] to speed up the Pollard rho method on elliptic or hyperelliptic curves with efficient automorphisms. Our main contribution is in giving a precise complexity analysis for the case when such feature is used to solve the DLP over finite fields of large extension degree.

The Pollard rho method on a cyclic group G traces a random walk over G by iteratively applying a function to G. The discrete logarithm of the target can be computed when a collision is reached in the random walk. When the iterating function is assumed to be random, a collision is expected to occur after about $\sqrt{|G|}$ applications of the iterating function, due to the birthday paradox. Thus the complexity of Pollard rho method is determined by the size of the base group G on which random walk is performed and the running time of the iterating function. Reducing either one of these two factors will result in speedup of the Pollard rho method.

The recent work [7] proposes an improvement of the Pollard rho method on prime fields by reducing the average time taken for computing each application of the iteration function. In contrast, our method restricts the random walk to a subset S of G so that a collision occurs after about $\sqrt{|S|}$ applications of the iterating function, which is smaller than $\sqrt{|G|}$. The main difficulty in doing this was finding a fast iterating function with an image set S that was much smaller than G. We make an explicit construction for subgroups of $\mathbf{F}_{p^m}^{\times}$, with the small prime and large extension degree case as our main target. Our construction yields speedup over the normal Pollard rho by a factor of $\frac{3p-3}{4p-3}\sqrt{m}$.

We also show how our method can be applied to a pairing-based cryptosystem, in which a bilinear map $e : G_1 \times G_2 \to G_T$ is used. When a Tate or Weil pairing is in use, G_1 is a subgroup of points on an elliptic curve $E(\mathbf{F}_{p^\ell})$ and G_T is a cyclic subgroup of $\mathbf{F}_{p^{k\ell}}^{\times}$, where k is the embedding degree. The MOV attack [13] transforms the DLP on G_1 or G_2 into a DLP on G_T. When our algorithm is applied to G_T, which is a subgroup of $\mathbf{F}_{p^{k\ell}}^{\times}$, the complexity of the DLP on G_T is reduced by a factor of $\frac{3p-3}{4p-3}\sqrt{k\ell}$. There are many cryptosystems [4, 11] that suggest the use of bilinear maps over binary or ternary fields, for computational and communication efficiency reasons. To be more explicit, $k\ell$ is set to 1132 and 726 in the binary and ternary curves suggested in [4, 11]. On these parameters, our method would give Pollard rho speedup by factors of 20.2 or 18.0, respectively.

We remark that, while we have achieved complexity lower than the straightforward application of Pollard rho, this does not conflict with the complexity lower bound known [18] for generic algorithms solving DLPs, as our method utilizes the *encoding* information for the group. When our construction is forcefully placed within the generic algorithm framework, the gain we achieve can be explained from the fact that some group operations are much easier than others, which is a property that is ignored in the generic algorithm framework. In fact, the result on generic algorithm complexity implies that, to achieve results better than normal Pollard rho, we should either exploit the group encoding or invent a method that mostly uses the fastest of the group operations.

Organization In Section 2, we introduce the Pollard rho method and its variants. In Section 3, we present a variant of Pollard rho, called the random walk restriction. In Section 4 and 5, we apply this method to finite fields of large extension degree and pairing-friendly elliptic curves over binary or ternary fields, respectively. Section 6 concludes this paper.

2 Pollard rho Algorithm

Throughout this paper $G = \langle g \rangle$ will be a finite cyclic group of prime order q, and $h = g^x$ will be the target we wish to find the discrete logarithm of. In this section, we briefly review some variants of the Pollard rho method.

Let us start by explaining the parts that are common to all the variants. Suppose we know $(a, b), (c, d) \in \mathbf{Z}_q \times \mathbf{Z}_q$, such that $g^a h^b = g^c h^d$ and $b \neq d$. Such a pair implies the linear equation $a + b \cdot x \equiv c + d \cdot x \pmod{q}$ and we can easily solve for x from this equation. All variants of the Pollard rho method suggest how to efficiently obtain such a double expression for a single element of G. The general strategy for obtaining the double expression is explained next.

We say that a function $f : G \to G$ is *exponent traceable*, or allows exponent tracing, with respect to g and h, if it is possible to express the function in the form

$$f(g^a h^b) = g^{f_g(a,b)} h^{f_h(a,b)},$$

for some easily computable functions $f_g, f_h : \mathbf{Z}_q \times \mathbf{Z}_q \to \mathbf{Z}_q$. Let us fix an exponent traceable function f and take a random $(a_0, b_0) \in \mathbf{Z}_q \times \mathbf{Z}_q$. We generate a sequence $(g_i)_{i \geq 0}$ through iterative applications of f, i.e., we set

$$g_0 = g^{a_0} h^{b_0} \quad \text{and} \quad g_{i+1} = f(g_i) \quad \text{for } i \geq 0.$$

Since G is finite, there must be integers $\mu \geq 0$ and $\lambda > 0$ satisfying $g_{\lambda+\mu} = g_\lambda$. The smallest of such integers are called the pre-period and period of the sequence $(g_i)_{i \geq 0}$, respectively. The exponent traceable property of f implies that we have access to the exponents (a_i, b_i) for each $g_i = g^{a_i} h^{b_i}$. Thus, an appropriate *collision* of the generated sequence, i.e., the event of $g_i = g_j$ with

$b_i \neq b_j$, allows us to compute the discrete logarithm of h, through the method described previously.

Below, we shall describe two exponent traceable iterating functions and some methods for detecting collisions. Any combination of the two components will produce a version of the Pollard rho algorithm.

2.1 Iterating functions

Because the value $\lambda + \mu$, called *rho length*, is expected to be $\sqrt{\pi q/2}$ when the function f is chosen uniformly at random from the set of all functions on G, an iterating function is considered to be of good design if its rho length is close to $\sqrt{\pi q/2}$.

There are two main types of iterating functions. One is the original *Pollard's iterating function* and the other is called *r-adding walks*. They are defined as follows. Let $G = T_0 \cup T_1 \cup T_2$ be a partition of G consisting of roughly equal sized subsets. The Pollard's iterating function f_P [15] is

$$f_P(y) = \begin{cases} gy, & \text{if } y \in T_1, \\ y^2, & \text{if } y \in T_2, \\ hy, & \text{if } y \in T_3. \end{cases}$$

Let r be a small positive integer and let $T_0 \cup \cdots \cup T_{r-1}$ be a partition of G into roughly the same sized subsets. For each $s = 0, \ldots, r-1$, set the multipliers $M_s = g^{m_s} h^{n_s}$ with randomly selected integers m_s, n_s. With the indexing function $s : G \to \{0, 1, ..., r-1\}$, that specifies to which $T_{s(y)}$ a given element $y \in G$ belongs to, the r-adding walk f_T is defined to be

$$f_T(y) = y \cdot M_{s(y)}.$$

The work [17] shows that the expected rho length for random walk sequences generated by an r-adding walk using any $r \geq 8$ is roughly of the order $\mathcal{O}(\sqrt{q})$ on any cyclic group. Number of tests [25] over elliptic curves have shown that the rho length of Pollard's original iterating function is larger than $\sqrt{\pi q/2}$, but that of 20-adding walks is very close to $\sqrt{\pi q/2}$.

2.2 Collision detection

The most naive approach to finding collisions is to store all generated points g_i until a collision occurs. The main issues with collision detection is to do this with minimal number of iterating function applications after the actual collision and with a small amount of memory. Recall that μ and λ denote the pre-period and period, respectively. Among the many collision detection methods proposed [5, 9, 14, 16, 23], we briefly explain those suggested by Floyd, Brent, and Quisquater-Delescaille.

Floyd The central idea is to wait for a collision of type $g_i = g_{2i}$ to happen. Three applications of the iterating function are needed at each iteration to update the two current states g_i and g_{2i} to g_{i+1} and $g_{2(i+1)}$, respectively. This will reach a collision within $\lambda + \mu$ iterations, or equivalently, $3(\lambda + \mu)$ applications of the iterating function. So, with a good iterating function, $3\sqrt{\pi q/2}$ applications are expected.

Brent We explain a method by Teske [24], which is an optimized version of the works [5, 19]. Eight most recent g_k, for which the index k are powers of 3, are kept in storage. After each application of the iterating function, the current g_i is compared with the eight stored entries, and g_i replaces the oldest of the eight entries if a new power of 3 has been reached. This will terminate with a collision between current g_i and some g_k from storage in $\{1.25 \cdot \max(\lambda/2, \mu) + \lambda\}$ iterations. When the iterating function is random, this is expected to be $1.229\sqrt{\pi q/2}$ iterating function applications.

Distinguished points [16] This was originally an idea for use with time-memory tradeoff techniques. Distinguished points are those elements of G that satisfy a certain condition, which is easy to check. For example, with a fixed encoding for G, we may set them to be those elements with a certain number of starting bits equal to zero.

After each application of the iterating function, the current g_i is added to a table, if and only if it is a distinguished point. The algorithm terminates when a collision is found among the stored distinguished points. The distinguished point should be defined so that this table is of manageable size.

Let θ be the fraction of elements in G which satisfy the distinguishing property. The algorithm is expected to terminate with a collision after $\sqrt{\pi q/2} + 1/\theta$ applications of the iterating function. This method has the advantage that it can lead to n-times speedup with n-processor parallelization [26].

3 Random Walk Restriction

In this section, we describe a general approach, previously applied to subgroups of elliptic curves [8, 12, 27], that results in faster DLP solving than straightforward applications of the Pollard rho variants. Recall that the complexity of Pollard rho variants is $\sqrt{\pi|G|/2}$ evaluations of iterating function when the cost of collision detection is ignored. This shows that two factors influence the complexity of Pollard rho variants. One is the complexity of iterating function evaluation and the other is the size of space on which the random walk is traversed. There are corresponding two approaches to the speedup of Pollard rho. One is the reduction of complexity for the evaluation of the iterating function and the other is the restriction of the random walks to a subset of G. The former was studied in [7] and the latter is the approach taken by this paper. After explaining the general method, which we shall call random walk restriction, its application to more concrete settings shall be explored in the next section.

3.1 Solving DLP with an iterating function of small image size

Fix any function $\varphi : G \to G$ which allows exponent tracing, but which may not be useful for solving DLP, i.e., the collisions generated are of the form $g^a h^b = g^c h^d$ with $b = d$. Let $f : G \to G$ be any exponent traceable iterating function suitable for solving DLP on G and consider the function $f_\varphi := \varphi \circ f$. It is clear that f_φ maps $\bar{G}$ to $\bar{G}$, where $\bar{G} = \mathrm{Im}(\varphi)$ is a notation we shall use throughout this section. Since f is always chosen to be exponent traceable, f_φ also satisfies the same condition. Notice that during the random walking of the Pollard rho method variants, we do not need G to be closed under the group operation. Thus, we will have no problem using f_φ as an iterating function on $\bar{G}$, which is a smaller set and which should bring earlier collision.

It remains to see if f_φ will show properties close to a random function. We have the following proposition to support this when φ is uniform in its number of pre-images.

Proposition 1. *Let $\varphi : G \to G$ be a function such that $|\varphi^{-1}(y)| = |G|/|\bar{G}|$ for all $y \in \bar{G} = \mathrm{Im}(\varphi)$. If f is selected uniformly at random from G^G, the set of all functions on G, then $f_\varphi = \varphi \circ f$ is a random function on $\bar{G}$.*

Proof. Consider the function $\tilde{\varphi} : G^G \to \bar{G}^{\bar{G}}$ mapping $f \mapsto (f_\varphi = \varphi \circ f)$. It suffices to show that $\tilde{\varphi}$ is uniform in its number of pre-images, so that each function on $\bar{G}$ is equally likely to be chosen, when each function f on G is equally likely to be chosen.

Let $\mathbf{y} = (y_j)_{j \in \bar{G}} \in \bar{G}^{\bar{G}}$ be any function on $\bar{G}$. We are viewing $\mathbf{y}$ as the function on $\bar{G}$ sending $j \mapsto y_j$. Suppose that $\mathbf{x} = (x_i)_{i \in G} \in G^G$ satisfies $\tilde{\varphi}(\mathbf{x}) = \mathbf{y}$. Then we must have $\varphi(x_j) = y_j$ for each $j \in \bar{G} \subset G$, but the equation $\tilde{\varphi}(\mathbf{x}) = \mathbf{y}$ places no restriction on x_i for $i \in G \setminus \bar{G}$. In fact, we have $\tilde{\varphi}(\mathbf{x}) = \mathbf{y}$ if and only if $\varphi(x_j) = y_j$ for each $j \in \bar{G}$. Since for each $j \in \bar{G}$, x_j may be any one of the $|G|/|\bar{G}|$-many elements of G, we have

$$|\tilde{\varphi}^{-1}(\mathbf{y})| = \Big(\frac{|G|}{|\bar{G}|}\Big)^{|\bar{G}|} \cdot |G|^{|G \setminus \bar{G}|},$$

which is evidently uniform over all $\mathbf{y} \in \bar{G}^{\bar{G}}$. □

Even though this proposition allows us to intuitively believe that f_φ on $\bar{G}$ is as good a random function on $\bar{G}$ as f is on G, as soon as we fix an iterating function f, the induced function f_φ on $\bar{G}$ is an explicit function, so the above proposition is no logical guarantee that f_φ will provide a good random walk. But this was the situation with the original f to begin with.

For a function f, let us denote its expected time for evaluation by $|f|$. The discussion so far allows us to state the main result of this section.

Theorem 1. *Let $\varphi : G \to G$ be an exponent traceable function. Then, given an exponent traceable iterating function f suitable for solving DLP on G and a collision detection method, by working with $f_\varphi = \varphi \circ f$ over $\bar{G} = \mathrm{Im}(\varphi)$, the expected time to solve the discrete logarithm problem over G can be reduced by a factor of $\sqrt{\frac{|\bar{G}|}{|G|}} \cdot \frac{|f_\varphi|}{|f|}$, compared to that of the original method over G.*

Notice that our result is mostly independent of which variant of Pollard rho algorithm we use. Use of the distinguished point method for collision detection is an exception, if one is to be very exact, but even this case can be dealt with by appropriately increasing its probability of appearance. As the expected rho length has decreased, this should not cause trouble with distinguished point storage requirements.

Remark 1. What can we expect of the random walk provided by f_φ, when φ is not uniform in its number of pre-images? This situation implies that during the walk provided by f_φ, some elements of $\bar{G}$ are more likely to occur than others. So we should arrive at a collision even earlier than expected of a random function. Such unevenness works to the DLP solver's advantage.

Example 1. Let $\omega \in \mathbf{Z}_q$ be of order t. Suppose φ is defined in such a way that it satisfies

$$\varphi(y) = \varphi(y^{\omega^i}), \qquad i = 0, 1, \dots, t-1,$$

in addition to being exponent traceable. Then the image of such a φ would be at most $\frac{1}{t}$-th of the total space. One possible candidate for such a function is to define $\varphi(y)$ as the minimum of $y, \dots, y^{\omega^{t-1}}$ under an appropriate ordering. In general, such a φ would have high evaluation complexity, i.e., t exponentiations and comparisons, but in a field of prime characteristic, φ can have low complexity. In Section 4, we follow this lines of reasoning.

It should be clear by now that we do not have to define f_φ as a composition of two functions. The essential idea is to design an iterating function, i.e., an exponent traceable function on G, with an image space that is much smaller than expected of random functions. In practice, it would be harder to design such a mapping directly than by composition.

4 Application to finite extension fields

In this section, we explain how the general theory is applied to a concrete case by suggesting an image-restricting function on large extension fields of small finite fields.

Let p be a prime and consider $\mathbf{F}_{p^m}$, the finite field of p^m elements. In this section, we apply results of the previous section to cyclic subgroups of $\mathbf{F}_{p^m}^\times$. We fix a normal basis $\{\alpha, \alpha^p, \dots, \alpha^{p^{m-1}}\}$ for $\mathbf{F}_{p^m}$ and write elements of $\mathbf{F}_{p^m}$ using the coordinates in the normal basis, i.e., write $\beta = b_0\alpha + \dots + b_{m-1}\alpha^{p^{m-1}}$ as $\beta = (b_0, ..., b_{m-1})$. As before, our objective is to solve for $\log_g h$ in a cyclic group $G = \langle g \rangle \subset \mathbf{F}_{p^m}^\times$ of prime order q.

There is a natural way to give $\mathbf{F}_p$ an ordering, and once a basis for $\mathbf{F}_{p^m}$ is fixed, we can give $\mathbf{F}_{p^m}$ the dictionary order using the ordering on $\mathbf{F}_p$. In particular, we have given an ordering to elements of $G \subset \mathbf{F}_{p^m}^\times$. We next define the map $\varphi : G \to G$ by

$$\varphi(y) = \min\{\, y^{p^i} \mid i = 0, \dots, m-1\}. \tag{1}$$

When $\mathbf{F}_{p^m}$ is encoded using the normal basis, the function φ outputs the smallest of all cyclic shifts of its input. Notice that in any realization of φ that uses the normal basis, the number i producing the minimum will be known, in which case φ is naturally exponent traceable. The exponent are simply multiplied by p^i.

To see the effects of the method given in Section 3.1, we need to compare the image size $|\bar{G}|$ with $|G|$. In the process we shall see that φ is almost uniform in its number of pre-images, but this information is not absolutely necessary for our purpose.

Proposition 2. *Let t be the smallest positive integer such that $q|p^t - 1$. Then $|\varphi^{-1}(y)| = t$ for every $y \in \mathrm{Im}(\varphi) \setminus \{\mathrm{id}\}$*

Proof. Let $y \in G\backslash\{\mathrm{id}\}$. It suffices to show that the number of distinct cyclic shifts of y is t. Suppose $y^{p^j} = y^{p^i}$ for some $0 \le j < i < m$. Writing this as $y^{p^i - p^j} = 1$ and recalling $|\langle y\rangle| = q$, we know $q|p^j(p^{i-j} - 1)$. But, as $\gcd(q, p^j) = 1$, we must have $q|p^{i-j} - 1$. Thus, the choice of t implies that $h, h^p, \cdots, h^{p^{t-1}}$ are all the distinct elements of G, and that this sequence is repeated for further powers. □

The proof of the above proposition has also shown that t is a divisor of m. If t is a proper divisor of m, then we are looking at the situation $G \subset \mathbf{F}_{p^t} \subset \mathbf{F}_{p^m}$. So, in any cryptographic application of $G \subset \mathbf{F}_{p^m}^{\times}$, we would have $t = m$, for, if otherwise, we would just be wasting resources. We state implications of this thought as a remark.

Remark 2. In any cryptographic application of $G \subset \mathbf{F}_{p^m}^{\times}$, the function φ of equation (1) is almost pre-image uniform in that the number of pre-images is $|\varphi^{-1}(y)| = m$ for every $y \in \mathrm{Im}(\varphi) \setminus \{\mathrm{id}\}$. Hence the size $|\mathrm{Im}(\varphi)|$ is very close to $|G|/m$.

So far, from discussions of Section 3.1, we know that, by working with f_φ over $\bar{G}$, the DLP over G can be solved with $\sqrt{1/m}$ factor reduction in the number of iterating function applications. To see how this translates into real advantage over the original direct approach on G, it remains to compare the speed of f to that of f_φ.

To evaluate the function φ given by (1), we need to find the smallest among m integers given as elements of $(\mathbf{F}_p)^m = \mathbf{F}_p \times \cdots \times \mathbf{F}_p$. This can be done with $m-1$ comparisons in $(\mathbf{F}_p)^m$. To compare $|\varphi|$ with $|f|$, we want to count this in terms of operations in $\mathbf{F}_p$.

Lemma 1. *The number of operations in $\mathbf{F}_p$ required to compare two integers in $(\mathbf{F}_p)^m$ is expected to be $\frac{p}{p-1}(1 - \frac{1}{p^m})$.*

Proof. Let $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{y} = (y_1, \ldots, y_m)$ be two integers from $(\mathbf{F}_p)^m$. If we had $x_1 = y_1$, $x_2 = y_2$, …, $x_{i-1} = y_{i-1}$, but $x_i \neq y_i$, then i comparisons in $\mathbf{F}_p$ would be required to find the smaller of $\mathbf{x}$ and $\mathbf{y}$. Thus our expected number of comparisons is

$$\sum_{i=1}^{m-1} i \cdot \left(\frac{1}{p}\right)^{i-1}\left(1 - \frac{1}{p}\right) \;+\; m \cdot \left(\frac{1}{p}\right)^{m-1},$$

where the last term is written separately because it looks slightly different. Evaluation of the sum results in our claim. □

This lemma shows that if p is large, the first comparison is likely to show us which of the two is smaller, and for even $p = 2$, two comparisons in $\mathbf{F}_2$ are enough. Thus we have the following lemma concerning the speed of φ.

Lemma 2. *On the cyclic group $G \subset \mathbf{F}_{p^m}^{\times}$, we can expect to evaluate the function φ given by equation* (1) *using $(m-1)\frac{p}{p-1}(1-\frac{1}{p^m})$ operations in $\mathbf{F}_p$.*

In general, evaluation of f is equal to one group operation in G. To compare $|\varphi|$ and $|f|$, we need to know the complexity of multiplication in $\mathbf{F}_{p^m}$ in normal basis representation.

To multiply two elements from a finite field under normal basis, classically, the matrix $T = (t_{i,j})$, defined by

$$\alpha^{p^i} \cdot \alpha^{p^j} = \sum_{0 \le k < m} t_{i-k,k-j} \alpha^{p^k}$$

is considered. For $\mathbf{x} = (x_0, ..., x_{m-1}),\ \mathbf{y} = (y_0, ..., y_{m-1}) \in \mathbf{F}_{p^m}$, product $\mathbf{xy}$ is computed as

$$\mathbf{z} = \mathbf{xy} = (z_0, ..., z_{m-1}),\ z_i = (x_i, ..., x_{i-1})\ T\ (y_i, ..., y_{i-1})^t,$$

where indexes are computed modulo m. Let d be the number of non-zero entries in T. Then multiplication in $\mathbf{F}_{p^m}$ can be done with at most $2md$ operations in $\mathbf{F}_p$ using T. Furthermore, we always have $d \ge 2m-1$ and better ways are known.

Let us say that a multiplication of two polynomials in $\mathbf{F}_p[x]$ of degree less than m can be done with at most $\mathsf{M}(m)$ operations in $\mathbf{F}_p$. Classical polynomial multiplication yields $\mathsf{M}(m) \le 2m^2$, but it is known [21, 22] that we may take $\mathsf{M}(m) \in \mathcal{O}(m \log m \log\log m)$.

At the moment, the fastest method for a multiplication under normal basis is one using Gauss periods [20]. The method for multiplication in $\mathbf{F}_{p^m}$ requires $\mathsf{M}(km) + (2k+1)m - 2$ operations in $\mathbf{F}_p$, for a positive number k corresponding to the type of Gauss period used. Thus, even if we disregard the more expensive $\mathsf{M}(km) \in \mathcal{O}(km \log(km) \log\log(km))$ part, whose multiplicative constant we are unaware of, we see that multiplication in $\mathbf{F}_{p^m}$ requires at least $3m$ operations in $\mathbf{F}_p$.

Putting together Lemma 2 and the above information, we can say that the time taken for φ evaluation is expected to be less than roughly $\frac{p}{3(p-1)}$ multiplications in $\mathbf{F}_{p^m}$. Since a usual iterating function involves one multiplication in $\mathbf{F}_{p^m}$, recalling $f_\varphi = \varphi \circ f$, we can state that $|f_\varphi| < \frac{4p-3}{3p-3}|f|$. The following is now a corollary to Theorem 1.

Theorem 2. *For a cyclic subgroup G of $\mathbf{F}_{p^m}^{\times}$, by using φ as given by equation* (1) *and by working with f_φ over* $\mathrm{Im}(\varphi) \subset G$*, we can speed up variants of the Pollard's rho algorithm by a factor greater than $\frac{3p-3}{4p-3}\sqrt{m}$.*

For example, on the cyclic subgroups of $\mathbf{F}_{2^{1024}}^{\times}$, assuming that the normal Pollard rho is faster than the index calculus method, we have $m = 1024$ and obtain complexity reduction of the DLP by a factor of at least $2^{4.26}$.

We end the explanation of our algorithm with a word of caution. When dealing with characteristic p fields, since $\varphi(h) = \varphi(h^p)$, any iterating function which utilizes p-th power cannot be used. In particular, with cyclic subgroups of the binary field, the original iterating function f_P, as suggested by the Pollard, cannot be used. This shows that one should pay attention to any glitches that could occur from interaction between φ and the iterating function.

Comparison with tag tracing The recent work [7] suggests an improvement of the r-adding walk called the tag tracing method for prime fields. It is also possible to apply the idea of [7] to finite extension fields in the normal basis representation, and we have explained this in the appendix.

Let us compare our random walk restriction and the tag tracing method on $\mathbf{F}_{p^m}$. With random walk restriction, the expected rho length is reduced by a $\sqrt{m}$-factor from that of the original r-adding walks while each step takes a little bit more time, i.e., $|f|+|\varphi|$, which is less than $\frac{4p-3}{3p-3}|f|$. On the other hand, in the tag tracing method, the rho length is preserved, but each step is replaced by a tag tracing step consisting of a tag computation, a table lookup, and an occasional computation of f. That is, random walk restriction reduces the number of steps taken by Pollard rho, while tag tracing decreases the effort taken by each step.

For the explicit example $\mathbf{F}_{2^{1024}}$, our random walk restriction achieves time reduction by a factor of $\frac{4p-3}{3p-3}\frac{1}{\sqrt{m}} = \frac{5}{96}$. Hence unless tag computation and table lookup combined requires less than $\frac{5}{96}|f|$ time, random walk restriction will be advantageous over tag tracing. On hardware implementations, where finite field multiplication can be relatively fast, table lookups may become the bottleneck and the goal of $\frac{5}{96}|f|$ per iteration will be hard to achieve with tag tracing. In addition, tag tracing requires a large storage space, which is not needed with random walk restrictions.

5 Application to Pairing Based Cryptosystem

The security of pairing based cryptosystem using a bilinear map $e : G_1 \times G_2 \to G_T$ depends on the DLP on G_1. For Weil or Tate pairings, G_1 is the order q subgroup of points in an elliptic curve $E(\mathbf{F}_{p^\ell})$ and G_T is a cyclic subgroup of $\mathbf{F}_{p^{k\ell}}^{\times}$. The positive integer k, called the *embedding degree* of G_1, is usually chosen so that it is difficult to solve DLP over $\mathbf{F}_{p^{k\ell}}^{\times}$ through the index calculus method. Since the MOV attack [13] reduces the DLP on G_1 to the DLP on a subgroup of G_T, it is anticipated that the best way to solve the DLP over G_1 is to use the Pollard rho method on G_1 or on the corresponding order q subgroup of G_T. We shall discuss the effect of our approach when it is applied to the subgroup of G_T.

Let us look at a specific example. The short signature scheme [4] uses supersingular curves $E^+(\mathbf{F}_{3^\ell}) : y^2 = x^3 + 2x + 1$ and $E^-(\mathbf{F}_{3^\ell}) : y^2 = x^3 + 2x - 1$. These curves have the embedding degree 6 and we have $G_T = \mathbf{F}_{3^{6\ell}}^\times$. Thus our method obtains a reduction of DLP complexity by a factor of $\frac{2}{3}\sqrt{6\ell}$ compared to what was originally expected from the given cyclic group. Hence, one should take care to use a larger cyclic subgroup of the elliptic curve than was previously used.

The table 1 was presented in [4], where DLog security refers to the bit size of largest prime divisor of $|E(\mathbf{F}_{3^\ell})|$. We have added a column giving corresponding values when the random walk restriction approach is considered, under the assumption that the complexity of elliptic curve addition is roughly the same as the complexity of finite field multiplication.

Curve	ℓ	DLog security ($\lceil \log_2 q \rceil$)	DLog security under our approach
E^-	79	126	118
E^+	97	151	142
E^+	121	155	146
E^+	149	220	211
E^+	163	256	247
E^-	163	259	250
E^+	167	262	253

Table 1. Security parameters for short signature

It is also noted in the paper [4] that due to the work [6], the DLP over $\mathbf{F}_{3^{6\ell}}^\times$ is easier than on prime fields of equivalent size, so that a large $\mathbf{F}_{3^{6\ell}}^\times$ should be taken. When this advice is followed, our method can work on the larger $\mathbf{F}_{3^{6\ell}}^\times$ to obtain a better reduction of DLP complexity.

Another example is the ID-based encryption [3]. Originally, these systems were built on elliptic curves over a large prime field $\mathbf{F}_p$ of embedding degree 2, on which our methods would yield no advantage. But for efficiency reasons Galbraith [11] suggested the use of elliptic curves over characteristic 2 or 3 fields with Tate pairing of embedding degrees 4 and 6, respectively. These curves are $y^2 + y = x^3 + x$ and $y^2 + y = x^3 + x + 1$ in characteristic 2 and $y^2 = x^3 + 2x \pm 1$ in characteristic 3. In each of these cases, our attack can be applied to reduce the complexity by a factor of $\frac{3}{5}\sqrt{4\ell}$ and $\frac{2}{3}\sqrt{6\ell}$, respectively. In practice, if one chooses $\ell = 283$ or 397 with the curve $y^2 + y = x^3 + x + 1$, we can speed up the Pollard rho algorithm by a factor of 20.2 or 23.9, respectively.

6 Conclusion

In this paper, we proposed a variant of the Pollard rho method, called random walk restriction, and showed that this idea can be applied to cyclic subgroups of finite fields of large extension degree. The main idea is to restrict the random walk of the Pollard rho to a smaller set which results in an earlier collision. As a result, our algorithm achieves speedup over the Pollard rho method by a factor of approximately $\frac{3p-3}{4p-3}\sqrt{m}$ in subgroups of $\mathbf{F}_{p^m}^{\times}$ and can be applied to pairing based cryptosystems.

Acknowledgments. Minkyu Kim and Jung Hee Cheon were supported by the Korea Science and Engineering Foundation (KOSEF) grant. (No. R01-2008-000-11287-0)

References

1. Memoirs of the 3rd Cryptology Paper Contest, arranged by a Korean government organization, 2007. (written in Korean)
2. L. Adleman, “A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography,” Proc. of the IEEE 20th Annual Symposium on Foundations of Computer Science (FOCS), pp.55-60, 1979.
3. D. Boneh and M. Franklin, “Identity-based encryption from the Weil pairing,” *Crypto ’01*, Springer, LNCS 2139, pp. 213-229, 2001.
4. D. Boneh, B. Lynn, and H. Shacham, “Short Signatures from the Weil Pairing,” *J. Cryptology*, Vol. 17, pp. 297-319, 2004.
5. R. Brent, “An improved Monte Carlo Factorization Algorithm,” *BIT*, Vol. 20, pp. 176-184, 1980.
6. D. Coppersmith, “Fast Evaluation of Logarithms in Fields of Characteristic Two,” *IEEE Trans. Inform. Theory*, Vol. 30(4), pp. 587-594, 1984.
7. J. Cheon, J. Hong, and M. Kim, “Speeding up Pollard Rho Method on Prime Fields,” *Asiacrypt ’08*, Springer, LNCS 5350, pp. 471-488, 2008.
8. I. Duursma, P. Gaudry, and F. Morain, “Speeding up the Discrete Log Computation on Curves with Automorphisms,” *Asiacrypt ’99*, LNCS 1716, Springer-Verlag, pp. 103-121, 1999.
9. D. Knuth, *The Art of Computer Programming, vol. II: Seminumerical Algorithms*, Addison-Wesley, 1969.
10. G. Frey and H.-G. Rück, “A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves,” *Math. Comp.*, Vol. 62, pp.865-874, 1994.
11. S. Galbraith, “Supersingular Curves in Cryptography,” *Asiacrypt 2001*, Springer, LNCS 2248, pp. 495-513, 2001.
12. R. Gallant, R. Lambert, and S. Vanstone, “Improving the Parallelized Pollard Lambda Search on Binary Anomalous Curves,” *Math. Comp.*, Vol. 69, pp. 1699-1705, 2000.
13. A. Menezes, T. Okamoto, and P. Vanstone, “Reducing elliptic curve logarithms to logarithms in a finite field,” *IEEE Trans. Inform. Theory*, Vol. 39(5), pp.1636-1649, 1993.
14. G. Nivasch, “Cycle Detection using a Stack,” *Information Processing Letters*, Vol. 90, pp. 135-140, 2004.
15. J. Pollard, “A Monte Carlo Method for Index Computation (mod p),” *Math. Comp*, Vol. 32(143), pp. 918-924, 1978.

16. J. Quisquater and J. Delescaille, “How easy is Collision Search? Application to DES,” *Eurocrypt ’89*, Springer-Verlag, LNCS 434, pp. 429-434, 1989.
17. J. Sattler and C. Schnorr, “Generating Random Walks in Groups,” *Ann.-Univ.-Sci.-Budapest.-Sect.-Comput.*, Vol. 6, pp.65-79, 1985.
18. V. Shoup, “Lower Bounds for Discrete Logarithms and Related Problems,” *Eurocrypt ’97*, Springer, LNCS 1223, pp. 256-266, 1997.
19. C. Schnorr and H. Lenstra, Jr., “A Monte Carlo Factoring Algorithm with Linear Storage,” *Math Comp.*, Vol. 43(167), pp. 289-311, 1984.
20. S. Gao, J. von zur Gathen, D. Panario and V. Shoup, “Algorithms for Exponentiation in Finite Fields,” *Journal of Symbolic Computation*, Vol. 29(6), pp.879-889, 2000.
21. A. Schönhage and V. Strassen, “Schnelle Multiplikation Grobner Zahlen,” *Computing*, Vol. 7, pp.281-292, 1971.
22. A. Schönhage, “Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2,” *Acta Informatica*, Vol. 7, pp.395-398, 1977.
23. R. Sedgewick, T. Szymanski, and A. Yao, “The Complexity of Finding Cycles in Periodic Functions,” *SIAM Journal on Computing*, Vol. 11, No. 2, pp. 376-390, 1982.
24. E. Teske, “Speeding up Pollard’s rho Method for Computing Discrete Logarithms,” *ANTS III*, Springer, LNCS 1423, pp. 541-554, 1998.
25. E. Teske, “On Random Walks for Pollard’s rho Method,” *Math. Comp.*, Vol. 70, pp. 809-825, 2001.
26. P. van Oorschot and M. Wiener, “Parallel Collision Search with Cryptanalytic Applications,” *J. Cryptology*, Vol. 12, pp. 1-28, 1999.
27. M. Wiener and R. Zuccherato, “Fast Attacks on Elliptic Curve Cryptosystems,” *Selected Areas in Cryptography ’98*, LNCS 1556, Springer-Verlag, pp.190-200, 1999.

Appendix: Tag Tracing Method with Normal Basis

In this appendix, we explain how tag tracing [7] can be applied to binary field with normal basis representation. Of course, this can then be easily adapted to any $\mathbf{F}_{p^m}$.

The tag tracing method is an improvement of the Pollard rho variant that uses r-adding walks. The main idea is to reduce the complexity of each iterating step. In the original r-adding walks, the iterative steps compute $g_{i+1} = g_i M_s$ from g_i, where $s = s(g_i)$ is the index of g_i computed by a function $s : G \to \{0, 1, \ldots, r-1\}$. This involves computing a full product at each step. But, if we have another function $\overline{s} : G \times G \to \{0, 1, \ldots, r-1\} \cup \{\text{fail}\}$ which computes the index of $g_i M_s$ without fully computing the product, we can reduce the execution time for each step as follows.

Let $\mathcal{M} = \{M_s = g^{m_s} h^{n_s}\}$. Fix a positive integer ℓ and prepare a precomputed table of $\mathcal{M}_\ell = (\mathcal{M} \cup \{1\})^\ell$. Given the i-th element $g_i \in G$ of the walk, compute the index $s_i = s(g_i)$. Without a full product of g_i and M_{s_i}, compute $s_{i+1} = \overline{s}(g_i, M_{s_i})$. Now, since $M_{s_i} M_{s_{i+1}} \in \mathcal{M}_\ell$, we can evaluate next index $s_{i+2} = \overline{s}(g_i, M_{s_i} M_{s_{i+1}})$. Continue this process until we arrive at ℓ iterations or need to store current g_j. It is easy to see that the execution time for each step is expected to be

$$|\overline{s}| + (\frac{1}{\ell} + P_{\text{fail}})|f|,$$

where P_{fail} is the probability of $\overline{s}$ evaluation to fail.

To apply the above procedure to $\mathbf{F}_{2^m}$ with the normal basis representation, define $s : G \to \{0, ..., r-1\}$ by

$$s(\mathbf{z}) = (z_0, z_1, \ldots, z_{\lceil \log r \rceil - 1}),$$

for $\mathbf{z} = (z_0, z_1, \ldots, z_{m-1}) \in \mathbf{F}_{2^m}$. Given $\mathbf{x}, \mathbf{y} \in \mathbf{F}_{2^m}$, since we can write their product as $\mathbf{xy} = (\mathbf{x}T\mathbf{y}^t, \ldots, \mathbf{x}^{p^{m-1}} T\big(\mathbf{y}^{p^{m-1}}\big)^t)$, we can compute $s(\mathbf{xy})$ with only $\lceil \log r \rceil \cdot m$ bit operations using pre-computed $T\mathbf{y}^t$, $T(\mathbf{y}^p)^t$, ..., $T(\mathbf{y}^{p^{\lceil \log r \rceil - 1}})^t$. This does not involve a full product of $\mathbf{x}$ and $\mathbf{y}$. Moreover, $P_{\text{fail}} = 0$.

Hence, the complexity of each steps of tag tracing is expected to

$$\lceil \log r \rceil \cdot m + \frac{1}{l} \cdot \mathsf{Mul},$$

where Mul denote the expected number of bit operations for multiplication in $\mathbf{F}_{2^m}$ using normal basis.