Speeding Up Elliptic Curve Discrete Logarithm Computations with Point Halving [⋆]

Fangguo Zhang and Ping Wang

School of Information Science and Technology,
Sun Yat-sen University, Guangzhou 510006, China
isszhfg@mail.sysu.edu.cn

Abstract. Pollard rho method and its parallelized variants are at present known as the best generic algorithms for computing elliptic curve discrete logarithms. We propose new iteration function for the rho method by exploiting the fact that point halving is more efficient than point addition for elliptic curves over binary fields. We present a careful analysis of the alternative rho method with new iteration function. Compared to the previous r-adding walk, generally the new method can achieve a significant speedup for computing elliptic curve discrete logarithms over binary fields. For instance, for certain NIST-recommended curves over binary fields, the new method is about 27% faster than the previous best methods in single-instance Pollard rho method. When running several instances of Pollard rho method concurrently, and computing the inversions using the simultaneous inversion algorithm by Peter Montgomery, the new method is about 12-17% faster than the previous best methods.

Keywords: Pollard rho method, elliptic curve discrete logarithm, point halving, random walk.

1 Introduction

Public-key cryptography based on elliptic curves over finite fields was proposed by Koblitz [22] and Miller [23] in 1985. Since then, elliptic curves over finite fields have been used to implement many cryptographic systems and protocols, such as the Diffie-Hellman key agreement scheme [2] [13], the elliptic curve variant of the Digital Signature Algorithm [1] [25], etc. In elliptic curve cryptography, the major security consideration is the intractability of the *elliptic curve discrete logarithm problem* (ECDLP).

Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$. Let $P \in E$ be a point of prime order n, and let $\langle P \rangle$ be the prime order subgroup of E generated by P. If $Q \in \langle P \rangle$, then $Q = kP$ for some integer k, $0 \leq k < n$. The problem of finding k, given P, Q, and the parameters of E, is known as the ECDLP. ECDLP

[⋆] This work is supported by the National Natural Science Foundation of China (No. 61070168).

is the foundation of the security of ECC, so solving it means breaking the ECC ciphers, therefore it has received a lot of attention.

For ECDLP, Pollard rho method and its modifications by Gallant, Lambert and Vanstone [16], and Wiener and Zuccherato [35] are to date known as the most efficient general algorithms. Van Oorschot and Wiener [34] showed that the modified Pollard rho method can be parallelized with linear speedup.

Pollard rho method is a randomized algorithm for computing discrete logarithms based on the Birthday Paradox. More precisely, an iteration function $F : G \to G$ is used to define a pseudo-random sequence Y_i by $Y_{i+1} = F(Y_i)$ for $i = 0, 1, 2, ...$, with some starting value Y_0. The sequence $Y_0, Y_1, Y_2, \ldots$ represents a walk in the group G. Because the order of the group is finite, the sequence will ultimately reach an element that has occurred before. This is called a *collision* or a *match*. The advantage of this method is that the space requirements can be small if one uses a clever method to detect the collision.

The basic assumption for the analysis of the expected run time of the rho method is that the walk Y_i behaves as a random walk. By this we mean that the iteration function $F : G \to G$ is a random mapping in the sense that for any $Y_i \in G$ the function F map Y_i to each element in G with the same probability $\frac{1}{|G|}$. However, in practice the iteration function $F : G \to G$ is not a truly random mapping, which always results in more iteration requirements. The problem of efficient simulation of a random walk in Pollard rho method is the central topic of this paper. Here, "efficient" means that the corresponding iteration function should require essentially no more than one group operation and use only constant or polynomial (in the input size of the ECDLP) storage.

In [32, 33], Teske proposed the efficient r-adding walk by applying more addition rules to the iteration function. Teske also proposed and analyzed the so called $r + s$-mixed walk, which introduce doubling steps to the r-adding walk. However, the $r + s$-mixed walk generally reduces certain randomness, and doubling is not much efficient than addition in the affine coordinate for elliptic curves. Therefore, the introduction of doubling steps does not lead to a significantly better performance.

Erik Knudsen [20] and Richard Schroeppel [30] independently proposed a new method for scalar multiplication of non-supersingular elliptic curves over binary fields. The idea is to replace all point doublings in double-and-add methods with a potentially faster operation called point halving. Knudsen [20] presented certain rough analysis which suggests that scalar multiplication with halvings could be 39% faster than scalar multiplication with doublings ([31] claims a 50% improvement). Bessalov [9] firstly used the idea of halving (he called division of points by two) to ECDLP, however, there was no detailed analysis for his approach.

In this paper, we design new iteration function for Pollard rho method by exploiting the fact that point halving is more efficient than point addition for elliptic curves over binary fields. We also present a careful analysis of the alternative rho method with new iteration function. Generally the new method can achieve a significant speedup for computing elliptic curve discrete logarithms over binary fields compared to the previous r-adding walk. Particularly, for certain NIST-recommended curves over binary fields, the new method is about 12-17% faster than the previous best methods.

The rest of this paper is organized as follows. We recall the Pollard rho method for elliptic curve discrete logarithm computation, and the concept and algorithms of point halving in section 2. In section 3, we propose the new iteration function, and present a careful analysis. We describe and analysis our experiments in section 4 and discuss the application to Koblitz curves in section 5. Finally, we conclude the paper in section 6.

2 Preliminaries

In this section, we recall the Pollard rho method for elliptic curve discrete logarithm computation, and discuss the concept and algorithms of point halving.

2.1 Pollard rho Method

Pollard [27] proposed an elegant generic algorithm for the discrete logarithms based on the Birthday Paradox and called it the rho method, which is an improvement over the well-known "baby-step giant-step" algorithm, attributed to Shanks [12]. Shanks' method allows one to compute discrete logarithms in a cyclic group G of order n in deterministic time $O(\sqrt{n})$ and space for $\sqrt{n}$ group elements. Pollard rho method also has time complexity $O(\sqrt{n})$ with only negligible space requirements; it is thus preferable.

Pollard rho method works by first defining a sequence of elements that will be periodically recurrent, then looking for a *match* in the sequence. The *match* will lead to a solution of the discrete logarithm problem with high probability. The two key ideas involved are the iteration function for generating the sequence and the cycle-finding algorithm for detecting a *match*.

If G is any finite set and $F : G \to G$ is a mapping and the sequence (X_i) in G is defined by the rule:

$$X_0 \in G,\ X_{i+1} = F(X_i)$$

this sequence is ultimately periodic. Hence, there exist unique integers $\mu \geq 0$ and $\lambda \geq 1$ such that $X_0, ..., X_{\mu+\lambda-1}$ are all distinct, but $X_i = X_{i+\lambda}$ for all $i \geq \mu$. A pair (X_i, X_j) of two elements of the sequence is called a *match* if $X_i = X_j$ where $i \neq j$. We call μ the *preperiod* and λ the *period* of the sequence (X_i). For the expected values of μ and λ, we have the following theorem:

Theorem 1. [19] *For any finite set D, under the assumption that an iteration function $F : D \to D$ behaves like a truly random mapping and the initial value X_0 is a randomly chosen group element, the expected values for μ and λ are $\sqrt{\pi|D|/8}$. The expected number of evaluations before a match appears is $E(\mu + \lambda) = \sqrt{\pi|D|/2} \approx 1.25\sqrt{|D|}$.*

Pollard [26] first applied this result to obtain an efficient and simple algorithm for factoring. Then in [27] he found an algorithm that uses the rho method to compute discrete logarithms in the multiplicative group $\mathbb{Z}_p^*$ (p prime) in the expected run time of $O(\sqrt{p})$ group operations. This algorithm can be easily generalized to compute discrete logarithms in arbitrary finite abelian groups, such as in groups of points of elliptic curves over finite fields.

Now we explain how the rho method for computing elliptic curve discrete logarithms works. Let P be a point of prime order n on an elliptic curve E over finite field, and let G be the subgroup of E generated by P. For any $Q \in G$, to compute k such that $Q = kP$, the rho method divides the group G into 3 subsets: S_1, S_2 and S_3 of roughly equal size, and defines the iteration function $F : G \to G$ as follows:

$$Y_{i+1} = F(Y_i) = \begin{cases} Y_i + P & Y_i \in S_1 \\ 2Y_i & Y_i \in S_2 \\ Y_i + Q & Y_i \in S_3 \end{cases}$$

Let the initial value $Y_0 = a_0P + b_0Q$ where a_0 and b_0 are two random numbers in $[0, n-1]$. Then each Y_i has the form $a_iP + b_iQ$, and the sequence (a_i) (and similarly for (b_i)) can be computed as follows:

$$a_{i+1} = \begin{cases} a_i + 1 \pmod n & Y_i \in S_1 \\ 2a_i \pmod n & Y_i \in S_2 \\ a_i \pmod n & Y_i \in S_3 \end{cases}$$

This implies that while computing (Y_i), we can easily keep track of the corresponding *sequences of exponents*, (a_i) and (b_i), such that $Y_i = a_iP + b_iQ$. Hence, as soon as we find a *match* (Y_i, Y_j), we have the following equation:

$$a_iP + b_iQ = a_jP + b_jQ$$

Since $Q = kP$, this gives

$$a_i + b_ik \equiv a_j + b_jk \pmod n$$

Now, if $\gcd(b_i - b_j, n) = 1$, we have $k = (a_j - a_i)(b_i - b_j)^{-1} \bmod n$.

Theorem 1 makes the assumption of true randomness. However, it has been shown empirically that this assumption does not hold exactly for Pollard iteration function [32]. The actual performance is worse than the expected value given in Theorem 1.

There are better iteration functions by applying more arbitrary multipliers. Assume that we are using r partitions (multipliers). We generate $2r$ random numbers,

$$m_j, n_j \in \{0, 1, \cdots, n-1\}, \text{ for } j = 1, 2, \cdots, r.$$

Then we precompute r multipliers $M_1, M_2, \cdots, M_r$ where,

$$M_j = m_jP + n_jQ, \text{ for } j = 1, 2, \cdots, r.$$

Define a hash function,

$$v : G \to \{1, 2, \cdots, r\}$$

Then the iteration function $F : G \to G$ defined as,

$$Y_{i+1} = F(Y_i) = Y_i + M_j, \text{ where } j = v(Y_i)$$

Then each Y_i has the form $a_i P + b_i Q$, and the sequences of (a_i) and (b_i) can be updated as follows,

$$a_{i+1} = a_i + m_j \pmod n \text{ and } b_{i+1} = b_i + n_j \pmod n.$$

This r-adding walk was first introduced in [29]. Sattler and Schnorr [28] showed that this approach is sufficiently random if $r \geq 8$. Teske [32] found experimentally that r-adding walk with $r \geq 20$ perform very close to a random walk.

The difference in performance between Pollard original walk and Teske's r-adding walk has been studied by [4]. We summarize the results as follows. In prime order subgroups of $\mathbb{Z}_p^*$, the value of $E(\mu + \lambda)$ for Pollard original walk and Teske's r-adding walk with $r = 20$ is $1.55\sqrt{|G|}$ and $1.27\sqrt{|G|}$, while in groups of points of elliptic curves over finite fields, the value is $1.60\sqrt{|G|}$ and $1.29\sqrt{|G|}$, respectively.

2.2 Point Halving

A non-supersingular elliptic curve E over $\mathbb{F}_{2^m}$ defined by the parameters $a, b \in \mathbb{F}_{2^m}, b \neq 0$, is the set of all solutions (x, y) to the equation

$$y^2 + xy = x^3 + ax^2 + b, \text{ where } x,\ y \in \mathbb{F}_{2^m},$$

together with an extra point $\mathcal{O}$, the *point at infinity.*

Let $H = (x_1, y_1) \in E$ be a point with $x_1 \neq 0$. Then $Q = 2H = (x_2, y_2)$ can be computed as follows:

$$x_2 = \lambda^2 + \lambda + a \tag{1}$$

$$y_2 = x_1^2 + (\lambda + 1)x_2 \tag{2}$$

$$\lambda = x_1 + \frac{y_1}{x_1} \tag{3}$$

Point halving is the reverse operation of point doubling: given $Q = (x_2, y_2)$, compute $H \triangleq \frac{1}{2}Q = (x_1, y_1)$ such that $Q = 2H$. One can compute point halving as follows: solve (1) for λ, (2) for x_1 and finally (3) for y_1. That is, solve $\lambda^2 + \lambda = x_2 + a$ for λ, and $x_1^2 = y_2 + (\lambda + 1)x_2$ for x_1, and finally compute $y_1 = x_1^2 + \lambda x_1$.

Let $|E(\mathbb{F}_{2^m})| = 2^k n$, where n is odd. Let P be a point of odd order n on E. It is obvious that point doubling and point halving are automorphisms of $\langle P \rangle$. Therefore, given a point $Q \in \langle P \rangle$, one can always find a unique point $H \in \langle P \rangle$ such that $Q = 2H$.

We say that the curve E has *minimal 2-torsion* if $k = 1$. Let $c \in \mathbb{F}_{2^m}$, we define the *trace* of c as $\mathrm{Tr}(c) = \sum_{i=0}^{m-1} c^{2^i}$. Then the property of E has minimal 2-torsion is equivalent to $\mathrm{Tr}(a) = 1$. Note that the NIST-recommended random curves [14] over binary fields have $\mathrm{Tr}(a) = 1$.

For the curve E with $\mathrm{Tr}(a) = 1$, one can compute point halving as follows. Solve the quadratic equation $\lambda^2 + \lambda = x_2 + a$ for λ. Let the two solutions be λ' and $\lambda' + 1$, one corresponds to λ and H and the other one to $\lambda + 1$ and $H + T$, where T is the point of order 2. In fact only H can be halved but not $H + T$. Therefore, λ' corresponds to λ and $H = (x_1, y_1)$ if and only if the equation $x^2 + x = x_1 + a$ has a solution in $\mathbb{F}_{2^m}$, that is, if and only if $\mathrm{Tr}(x_1 + a) = 0$. Further, we have $\mathrm{Tr}(x_1 + a) = \mathrm{Tr}(x_1^2 + a^2)$.

Hence, one first computes $w = x_2(\lambda' + 1) + y_2$, which is a candidate for x_1^2. If $\mathrm{Tr}(w + a^2) = 0$ then $\lambda = \lambda'$ and $x_1 = \sqrt{w}$. Otherwise, $\lambda = \lambda' + 1$ and $x_1 = \sqrt{w + x_2}$. More precisely, we summarize the above steps in the following Algorithm 1 [3].

Algorithm 1 Point Halving

```
Input: $Q = (x_2, y_2) \in \langle P \rangle$.
Output: $H = (x_1, y_1) \in \langle P \rangle$, where $Q = 2H$.
1: compute $\lambda$ such that $\lambda^2 + \lambda = x_2 + a$.
2: $w \leftarrow x_2(\lambda + 1) + y_2$.
3: if $\mathrm{Tr}(w + a^2) = 0$ then
4:     $x_1 \leftarrow \sqrt{w}$, $y_1 \leftarrow x_1(x_1 + \lambda)$.
5: else
6:     $x_1 \leftarrow \sqrt{w + x_2}$, $y_1 \leftarrow x_1(x_1 + \lambda + 1)$.
7: end if
```

Note that to make use of point halving in the iteration function for ECDLP computations, we will focus on the affine representation of point in point halving rather than the λ-representation mentioned in [15]. Further, one can generalize Algorithm 1 to the case of curve E with $\mathrm{Tr}(a) = 0$, that is $|E(\mathbb{F}_{2^m})| = 2^k n$ with $k > 1$ and n odd. In this case [20], it is necessary to solve k equations, perform $k + 1$ multiplications, one test, and k or $k + 1$ square root computations to find (x_1, y_1).

Fong *et al.* [15] provided a careful analysis of the actual efficiency of point halving for elliptic curves over binary fields with polynomial basis, such as the NIST-recommended random binary curves over $\mathbb{F}_{2^m}$ [14]. We summarize the results as follows. Let M, S and I denote the cost of field multiplication, squaring and inversion respectively. Then experimentally, the cost of solving the quadratic equation is approximately in the range $\frac{1}{2}M$ to $\frac{2}{3}M$, and the cost of computing square roots in $\mathbb{F}_{2^m}$ is expected to be in the range $\frac{1}{8}M$ to $\frac{1}{2}M$. As a result, the cost of a point halving with affine representation is roughly in the range of $[\frac{21}{8}M, \frac{19}{6}M]$. While point addition and doubling in affine coordinates need approximately the same costs: $I + 2M + S$. A careful analysis of the software implementation of multiplication and inversion in $\mathbb{F}_{2^m}$ is necessary for a fair comparison of halving and addition. Extensive experiments from [15] suggest that a realistic estimate of the ratio I/M of inversion to multiplication cost is 8 (or higher). Thus the cost of point addition or doubling in affine coordinates is generally larger than $10M$.

3 New Alternative Iteration Function

Iterative evaluations are the main operations of the Pollard rho method. We focus on how to design efficient iteration function in this section. For efficiency, generally we have the following criteria: a) for each iteration, the corresponding iteration function $F : G \to G$ should require essentially no more than one group operation, b) the iteration function F behaves like or close to a truly random mapping and c) the method use only constant or polynomial (in the input size of the ECDLP) storage.

3.1 Iteration Function

Since point halving is much more efficient than point addition for elliptic curve over binary fields, we can introduce point halving into the random walk, replace certain point additions with point halvings, to speed up the iteration for the rho method. Therefore we propose the following $r + h$-mixed walk.

Let P be a point of prime order n on an elliptic curve E over binary field, and let G be the subgroup of E generated by P. For any $Q \in G$, to compute k such that $Q = kP$, we divide the group G into $r + h$ subsets: $S_1, S_2, \ldots, S_r, S_{r+1}, \ldots,$ and S_{r+h} of roughly equal size, and we generate $2r$ random numbers,

$$m_j, n_j \in \{0, 1, \cdots, n-1\}, \text{ for } j = 1, 2, \cdots, r.$$

Then we precompute r multipliers $M_1, M_2, \cdots, M_r$ where,

$$M_j = m_j P + n_j Q, \text{ for } j = 1, 2, \cdots, r.$$

Define a hash function,

$$v : G \to \{1, 2, \cdots, r + h\}.$$

The iteration function $F : G \to G$ is called $r + h$-mixed walk if F defined as,

$$Y_{i+1} = F(Y_i) = \begin{cases} Y_i + M_{v(Y_i)} & v(Y_i) \in \{1, \ldots, r\} \\ \frac{1}{2} Y_i & v(Y_i) \in \{r+1, \ldots, r+h\} \end{cases}$$

Let the initial value $Y_0 = a_0 P + b_0 Q$ where a_0 and b_0 are two random numbers in $[0, n-1]$. Then each Y_i has the form $a_i P + b_i Q$, and the sequence (a_i) and

(b_i) can be computed as follows,

$$a_{i+1} = \begin{cases} a_i + m_{v(Y_i)} & \pmod n \quad v(Y_i) \in \{1, \dots, r\} \\ \frac{1}{2} a_i & \pmod n \quad a_i \text{ even and } v(Y_i) \in \{r+1, \dots, r+h\} \\ \frac{1}{2}(a_i + n) & \pmod n \quad a_i \text{ odd and } v(Y_i) \in \{r+1, \dots, r+h\} \end{cases}$$

$$b_{i+1} = \begin{cases} b_i + m_{v(Y_i)} & \pmod n \quad v(Y_i) \in \{1, \dots, r\} \\ \frac{1}{2} b_i & \pmod n \quad b_i \text{ even and } v(Y_i) \in \{r+1, \dots, r+h\} \\ \frac{1}{2}(b_i + n) & \pmod n \quad b_i \text{ odd and } v(Y_i) \in \{r+1, \dots, r+h\} \end{cases}$$

Correspondingly, once we find a *match* (Y_i, Y_j), we have the following equation:

$$a_i P + b_i Q = a_j P + b_j Q$$

Then, if $\gcd(b_i - b_j, n) = 1$, we have $k = (a_j - a_i)(b_i - b_j)^{-1} \bmod n$.

It is clear that the randomness of Teske's r-adding walk is better than that of the original Pollard walk, and the performances of both the r-adding walk and original walk are worse than what would expected by a truly random process [32]. Further, Teske [32] has given experimental evidence that introduce certain doubling rules into r-adding walk may lead to poor performance. Since point halving is the reverse operation of point doubling, we conjecture that the expected time for the proposed new random walk reach a collision is worse than what would be expected by r-adding walk.

The original Pollard rho method, the r-adding walk and the new $r+h$-mixed walk for finding discrete logarithms are based on a pseudo-random approximation to a Markov chain on a cycle group G. For each step of the new iteration, we have $Y_i = a_i P + b_i Q = (a_i + b_i k)P$, where a_i and b_i are updated by the above rules. Further, we define the sequence (u_i) by

$$u_i = a_i + b_i k \pmod n. \tag{4}$$

Thus the mapping,

$$Y_i \in G \mapsto u_i \in \mathbb{Z}_n,$$

is a one-to-one bijection between the new random walks (Y_i) on G and the walks (u_i) on the integers mod n. Since we want to produce sequences (Y_i) with expected preperiods and periods as close as possible to the case of the truly random walk, it is desirable that the distributions of the u_i generated by our iteration function get as close as possible to the uniformly distributed on the integers mod n.

Therefore, instead of studying the new random walks on G, we may restrict ourselves to walks of the form (4). Assuming v is a random hash function, then the corresponding walk (u_i) is a random walk on $\mathbb{Z}_n$, and such walks have been extensively studied in the literature.

3.2 Analysis

To estimate the running time until a collision is reached, we make the heuristic assumption that the above hash function $v : G \to \{1, 2, \cdots, r+h\}$ is a sufficiently random mapping.

Certain heuristics for analyzing the randomness in the iteration function of Pollard rho method have been widely discussed, and one can refer to the papers [6, 8, 10, 11] for details. In fact, point halving is the reverse operation of point doubling, therefore, the previous result can be easily applied to the new iteration function.

The new iteration function is a mixed iteration function: every step maps an intermediate value Y_i to $Y_{i+1} = F_k(Y_i)$, where $k \in \{1, 2, \ldots, r+h\}$. Let p_i be the probability that the ith rule is used with $1 \leq i \leq r$, and p_H be the probability that the point halving is used, and $p_1 + p_2 + \ldots + p_r + p_H = 1$. Then the average number of iterations before the first collision is reached is approximately

$$\sqrt{\frac{\pi n}{2(1 - p_H^2 - \sum_{i=1}^{r} p_i^2)}}.$$

That is, for the random walk that maps Y to $F_i(Y)$ with probability p_i and p_H as above, the reductions of randomness increase the expected number of iterations by a factor of $1/\sqrt{1 - p_H^2 - \sum_{i=1}^{r} p_i^2}$, which is very close to 1 in most cases. There is a trade-off between the use of point halving and the randomness of the new walk. That is, the more point halvings we use, which leads to more efficient iteration function, the worse the randomness of the walk. Hence, we need to find the optimal ratio that point halvings should account for to achieve the best performance.

For instance, the cases $(r, h) = (20, 20)$ and (128, 128) correspond to using halving with probability $\frac{1}{2}$ and using addings with probability $\frac{1}{40}$ and $\frac{1}{256}$ respectively. The factor in these cases is 1.164 and 1.155 respectively. Moreover, this value tends to $\frac{2}{\sqrt{3}} = 1.154$ as r goes to infinity. Hence, one expects a walk that uses halvings with probability $\frac{1}{2}$ should require about 1.154 many group operations as only adding walks, when one uses very large values for r and works in very large groups (so that other effects that influence the "randomness" of walks are less significant).

Consider the NIST-recommended random curves [14] over binary fields $\mathbb{F}_{2^{233}}$, with reduction trinomial $f(x) = x^{233} + x^{74} + 1$. Experimentally, the cost of a point halving with affine representation needs approximately $\frac{21}{8}M$. Under the assumption that the ratio I/M of inversion to multiplication cost is 8, point addition in affine coordinates needs more than $10M$. Compare the $r + h$-mixed walk (set $r = 20, h = 20$) with the r-adding walk, it is obvious that the new method is about

$$\frac{10 - 1.154 * (\frac{1}{2} * 10 + \frac{1}{2} * \frac{21}{8})}{10} \approx 27\%$$

faster than the r-adding walk. On the other hand, from a pessimistic point of view, if the cost of a point halving with affine representation needs $\frac{19}{6}M$, then the new method is about 24% faster than the r-adding walk. Generally, one may choose the ratio $\frac{h}{r+h}$ according to the actual cost of point halving and point addition over certain binary fields.

In practice, most typical choice when instantiating Pollard rho method to solve ECDLP is to run several walks in parallel and to compute the inversions using the simultaneous inversion algorithm by Peter Montgomery [24]. We follow standard practice of handling m iterations in parallel and batching mI into $1I + (3m-3)M$, where m is the number of processors. Then m point additions processed in parallel costs $1I+(5m-3)M+mS$. Hence, the average cost of point addition becomes $5M + S + \frac{1}{m}(I - 3M)$. Note that, with a polynomial basis, according to [17], $S \approx 1/7.5M$ for $\mathbb{F}_{2^{163}}$ and $1/9M$ $\mathbb{F}_{2^{233}}$. Taking into account the costs of memory operations and communications among processors, the average cost of point addition is roughly equal to $6M$. In this case, the new method is about

$$\frac{6 - 1.154 * (\frac{1}{2} * 6 + \frac{1}{2} * \frac{21}{8})}{6} \approx 17\%$$

faster than the r-adding walk. However, by some simple calculations it is clear that the optimal choice in this case is to set $p_H = 0.56$, then the new method is about

$$\frac{6 - \frac{1}{\sqrt{1-p_H^2}} * ((1-p_H) * 6 + p_H * \frac{21}{8})}{6} \approx 17.3\%$$

faster than before. Correspondingly, if the cost of a point halving needs $\frac{19}{6}M$, then the new method is about 12% faster than before with $p_H = \frac{1}{2}$.

4 Experiments

To explore the optimal performance for the new random walk, we implement the alternative rho using the new iteration functions with different parameters, and compare their performances with the r-adding walk. In this section, we describe these experiments and analysis the results.

For our experiments, we briefly introduce the elliptic curve group over binary fields and the notation we use in the following. Let $a, b \in \mathbb{F}_{2^m}, b \neq 0$. Then the elliptic curve $E_{a,b}$ over $\mathbb{F}_{2^m}$ is defined through the equation

$$E_{a,b} : y^2 + xy = x^3 + ax^2 + b, \text{ where } x,\ y \in \mathbb{F}_{2^m}.$$

Let $P \in E_{a,b}(\mathbb{F}_{2^m})$ be a point of prime order n, let G denote the subgroup of $E_{a,b}$ generated by P. Given $Q \in G$, determine the integer $0 \leq k < n$ such that $Q = kP$.

Let $W = (x, y)$ be any point of G, here we interpret x as an integer. We define the partition of G into $r + h$ subsets $S_1, S_2, \cdots, S_{r+h}$ as follows. First we compute a rational approximation A of the golden ratio $(\sqrt{5}-1)/2$, with a precision of $2 + \lfloor \log_{10}(q(r+h)) \rfloor$ decimal places. Let

$$v^* : G \to [0, 1), \quad (x, y) \to \begin{cases} Ax - \lfloor Ax \rfloor & \text{if } W \neq \mathcal{O} \\ 0 & \text{if } W = \mathcal{O} \end{cases} \tag{5}$$

where $Ax - \lfloor Ax \rfloor$ is the non-negative fraction part of Ax. Then let

$$v : G \to \{1, 2, \cdots, r+h\}, \quad v(W) = \lfloor v^*(W) \cdot (r+h) \rfloor + 1$$

and

$$S_i = \{W \in G : v(W) = i\}$$

This method is originally from Knuth's multiplicative hash function [21] and suggested by Teske [33]. From the theory of multiplicative hash functions we know that among all numbers between 0 and 1, choosing A as a rational approximation of $(\sqrt{5} - 1)/2$ with a sufficiently large precision leads to the most uniformly distributed hash values, even for non-random inputs.

Algorithm 2 Experiments for the r-adding walk and the new $r+h$-mixed walk

```
Input: Different iteration functions $F : G \to G$ (the $r$-adding walk and the new $r+h$-mixed walk with different $r$ and $h$).
Output: The average ratio $R =$ (number of steps until reach the collision)$/\sqrt{n}$.
 1: for $i = 30$ to $32$ do
 2:   for $j = 1$ to $20$ do
 3:     $m \leftarrow i + 1$.
 4:     repeat
 5:       Choose two random numbers $a, b \in \mathbb{F}_{2^m}$, where $b \neq 0$ and $\mathrm{Tr}(a) = 1$.
 6:       $n \leftarrow \frac{|\mathbb{E}_{a,b}|}{2}$.
 7:     until $n$ is prime and $2^i < n < 2^{i+1}$
 8:     Choose a random point $W \in \mathbb{E}_{a,b}$, where the order of $W$ equal to $|\mathbb{E}_{a,b}|$.
 9:     $P \leftarrow 2W$ (then $P$ is the generator of $G$).
10:     $t \leftarrow \frac{40000}{2^{i-30}}$.
11:     for $l = 1$ to $t$ do
12:       Choose a random number $c \in [0, n-1]$, $Q \leftarrow c * P$.
13:       Choose a random point in $G$ be the initial point $Y_0$.
14:       $k \leftarrow 1$.
15:       repeat
16:         $Y_k \leftarrow F(Y_{k-1})$.
17:         Check whether $Y_k$ is a distinguished point.
18:       until Reach the collision
19:       $R_l \leftarrow k/\sqrt{n}$.
20:     end for
21:     $R_j \leftarrow (\sum_{l=1}^{t} R_l)/t$.
22:   end for
23:   $R_i \leftarrow (\sum_{j=1}^{20} R_j)/20$.
24: end for
```

The purpose of our experiments is to evaluate the expected numbers of steps until a match is found using the r-adding walk and the new $r+h$-mixed walk with different parameter settings for r and h. Generally, we set the integer m in certain range. Then we randomly choose the parameters a and b where $a, b \in \mathbb{F}_{2^m}$ and $\mathrm{Tr}(a) = 1$, which determine the unique elliptic curve $\mathbb{E}_{a,b}$ over $\mathbb{F}_{2^m}$. We will check whether $\frac{|\mathbb{E}_{a,b}(\mathbb{F}_{2^m})|}{2}$ is a prime number within a certain range. If not, repeat the above procedures until we get a prime order subgroup G of $\mathbb{E}_{a,b}(\mathbb{F}_{2^m})$ with minimal 2-torsion. Then we set the generator P of G and choose a random point Q of G. When using the r-adding walk and the new $r+h$-mixed walk with different r and h to compute this discrete logarithm, we count the number of steps we perform until find a match. Then we determine the ratio R of the number of steps and $\sqrt{n}$. We repeat these steps a couple of times with the same P but several randomly chosen Q. Furthermore, for practical reasons, we do the above procedures with a couple of groups where the group order n between 2^{30} and 2^{32}. Therefore, We have Algorithm 2.

More precisely, for each $i \in [30, 32]$, we generate 20 elliptic curves, where each of them have a subgroup G of prime order n, such that $n \in [2^i, 2^{i+1}]$. Then for each group G, we generate 10000 to 40000 DLPs with the same generator P but randomly generated Q. The number of elliptic curves and instances of DLPs computed is given in Table 1. For each DLP, we use Teske's r-adding walk and the new $r+h$-mixed walk for iteration function, and find the match using distinguished point method. Once reaching a match, we compute the ratio R_l as (the number of steps until match is found)/$\sqrt{n}$. Then we compute the average ratio R_j of all DLPs over the same elliptic curve. Finally, we count the average ratio R of all DLPs with the same i, where $i \in [30, 32]$ and $n \in [2^i, 2^{i+1}]$.

Table 1. Number of elliptic curves and instances of DLPs

Bits	#Elliptic Curves	#DLPs per Curve
30	20	40000
31	20	20000
32	20	10000

Now, let us explain the parameters for distinguishing property in more detail. For example, if $i = 32$, which means n, the order of G, is a 32 bits prime number. According to [32], we are expected to take $1.292\sqrt{n}$ iterations before reaching a match. We define the distinguishing property as the Hamming weight of x-coordinate of the point less than or equal to 9. Each point has probability almost exactly $(\binom{32}{0} + \binom{32}{1} + \cdots + \binom{32}{9})/2^{32} \approx 2^{-6.64}$ of being a distinguished point. That is, to find a collision it is expected to compute $1.292 * 2^{-6.64}\sqrt{n}$ distinguished points.

For the r-adding walk, we set $r = 20$ and $r = 128$ (with h = 0), the typical value suggested by the literature, and for the new $r+h$-mixed walk, we set $r = 20$, $h = 10$; $r = 20$, $h = 20$; $r = 128$, $h = 64$; $r = 128$, $h = 128$ and $r = 64$, $h = 128$; respectively. The experiment results are given in Table 2. It shows that introducing point halving into the iteration rules indeed reduce the randomness of the random walk, consistent with our conjecture. However, the efficient point halving computation still improve the whole performance of the alternative rho by properly setting parameters r and h. Moreover, Table 2 also shows that set $r = 128$ rather than 20 does not lead to a significantly better performance.

Table 2. Performances for the r-adding walk and the new $r + h$-mixed walk

Bits	#DLPs	r	20	20	20	128	128	128	64
		h	0	10	20	0	64	128	128
30	800000		1.301	1.349	1.397	1.297	1.350	1.403	1.833
31	400000		1.280	1.352	1.414	1.291	1.354	1.401	1.842
32	200000		1.296	1.357	1.401	1.284	1.349	1.392	1.919
Average	1400000		1.294	1.351	1.402	1.293	1.351	1.401	1.848

5 Application to Koblitz Curves

In fact, besides the general elliptic curves over binary fields, the new method can also be applied to speed up computations of ECDLP on Koblitz curves with Frobenius endomorphism.

For Koblitz curves, the map $\phi : E(\mathbb{F}_{2^m}) \to E(\mathbb{F}_{2^m})$ defined by $\phi(x, y) = (x^2, y^2)$ is called the Frobenius endomorphism. There exists an integer λ such that $\phi(P) = \lambda P$ for all points P in G. Hence, one can define the equivalence relation $\sim$ by combining the Frobenius endomorphism and the negation map to get a speedup of $\sqrt{2m}$ [16] [35]. We denote the set of equivalence classes by $E/\sim$.

Based on Harley's work on ECC2K-95/ECC-2K108 [18] and [16], Bailey *et al.* [6] proposed an efficient and practical alternative iteration function for the rho method as follows,

$$Y_{i+1} = Y_i + \phi^l(Y_i) \tag{6}$$

where l is a function defined on the equivalence classes, which ensure that points from the same equivalence class have the same value l. For example, [6] define it as $l = ((\mathrm{HW}(x_{Y_i})/2) \bmod 8) + 3$, where $\mathrm{HW}(x_{Y_i})$ is the Hamming weight of the x-coordinate of Y_i. The variant iteration function is well defined on equivalence classes, and can combine with the distinguished points technique to achieve a speedup of $\sqrt{2m}$.

Note that to make use of Frobenius endomorphism, one need define the iteration function via the normal basis representation [6] [18]. Let $|E(\mathbb{F}_{2^m})| = 2^k n$ with $k \geq 1$ and n odd. For Koblitz curves with $k = 1$ (have minimal 2-torsion), the computation of point halving in normal basis is even more efficient than in polynomial basis. Let $\{\alpha, \alpha^2, \ldots, \alpha^{2^{m-1}}\}$ be a normal basis of $\mathbb{F}_{2^m}$. The trace of an element $c = \sum c_i \alpha^{2^i} = (c_{m-1}, \ldots, c_0)$ is given by $\mathrm{Tr}(c) = \sum c_i$. The square root computation is a right rotation: $\sqrt{c} = (c_0, c_{m-1}, \ldots, c_1)$. Squaring is a left rotation, and $x^2 + x = c$ can be solved bitwise. The cost of these operations can be neglected compared to field multiplication, so, the computation of point halving needs only about one field multiplication.

Therefore, for ECDLP over Koblitz curves, we propose the following $r+h$-mixed walk to achieve a further speedup. Let $l = v(Y_i)$, the iteration function $F : E/\sim \to E/\sim$ defined as,

$$Y_{i+1} = F(Y_i) = \begin{cases} Y_i + \phi^l(Y_i) & l \in \{1, \dots, r\} \\ \frac{1}{2}Y_i & l \in \{r+1, \dots, r+h\} \end{cases} \tag{7}$$

where v is a function defined on the equivalence classes. Then, each Y_i has the form $a_iP + b_iQ$, and the indices a_i and b_i can be updated correspondingly. Now, we explain why the new map is a well-defined map on $E/\sim$. More precisely, we have the following theorem.

Theorem 2. *Let $\sim$ be the equivalence relation and F be the random mapping on $E/\sim$ defined as above. If $Y_i \sim Y_j$ for certain integers i and j, then $Y_{i+1} \sim Y_{j+1}$. Moreover, if $Y_i = \phi^l(Y_j)$ for certain integers i, j and l, then $Y_{i+1} = \phi^l(Y_{j+1})$; and if $Y_i = -Y_j$, then $Y_{i+1} = -Y_{j+1}$.*

Proof. If $Y_i \sim Y_j$, then according to the definition of the equivalence relation, there exist certain integers i, j and l, such that $Y_i = \phi^l(Y_j)$. Let $k = v(Y_i) = v(Y_j)$. If $k \in \{1, \dots, r\}$, we have

$$Y_{i+1} = Y_i + \phi^k(Y_i) = (1+\lambda^k)Y_i = (1+\lambda^k)\lambda^l Y_j.$$

Also, we know

$$Y_{j+1} = Y_j + \phi^k(Y_j) = (1+\lambda^k)Y_j.$$

then

$$\phi^l(Y_{j+1}) = (1+\lambda^k)\lambda^l Y_j.$$

That is, $Y_{i+1} = \phi^l(Y_{j+1})$ and $Y_{i+1} \sim Y_{j+1}$.

On the other hand, if $k \in \{r+1, \dots, r+h\}$, we have

$$Y_{i+1} = \frac{1}{2}Y_i = \frac{1}{2}\lambda^l Y_j.$$

and

$$\phi^l(Y_{j+1}) = \phi^l(\frac{1}{2}Y_j) = \frac{1}{2}\lambda^l Y_j.$$

Therefore, $Y_{i+1} = \phi^l(Y_{j+1})$ and $Y_{i+1} \sim Y_{j+1}$. Further, if $Y_i = -Y_j$, it is trivial to check that $Y_{i+1} = -Y_{j+1}$. $\square$

Theorem 3 shows that once the random walk defined by the new iteration function falls into the same equivalence class with certain previous value, from that value on, the current walk and the previous walk will always fall into the same equivalence class for each step. This feature is the key point for the variant Pollard rho works. Then we can combine this feature with the distinguished point method to detect the collision. Therefore, the new iteration function is a well-defined map on $E/\sim$.

Correspondingly, by introducing point halving into the iteration on equivalence classes, one can expect to achieve certain further speedup for computing ECDLP over Koblitz curves. In fact, our theoretical estimate of the speedups in section 3.2 also applies to the Koblitz curves case.

We studied the bit-slicing technique [5–7], as mentioned in [5], binary Edwards curves do not appear to save time for ECC2-X, and the implementation of bit-slicing uses standard affine coordinates for Weierstrass curves. The bit-slicing technique is adopted to speedup the multiplication of the underlying field, and correspondingly the inversion operation. However, it does not change the assumption of the ratio I/M of inversion to multiplication. We confirm that there is no obstacle for the new method work with the bit-slicing technique.

6 Conclusion

In this paper, we adapt a new iteration function for the rho method to allow efficient use of point halving for computing elliptic curve discrete logarithms over binary fields. We discuss the new algorithm in theoretical analysis and propose certain practical settings. Compare to the previous r-adding walk, generally the new $r + h$-mixed walk can achieve a significant speedup for computing elliptic curve discrete logarithms over binary fields. Particularly, in the case of certain NIST-recommended curves over binary fields the new approach is about 12-17% faster than the previous best methods. We also show that the new approach can be applied to Koblitz curves.

References

1. ANSI X9.62-199x: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), January 13, 1998.
2. ANSI X9.63-199x: Public Key Cryptography for the Financial Services Industry: Elliptic Curve Key Agreement and Transport Protocols, October 5, 1997.
3. R. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren, "Handbook of Elliptic and Hyperelliptic Curve Cryptography". CRC Press, 2005.
4. S. Bai and R. P. Brent, "On the efficiency of Pollard's rho method for discrete logarithms", CATS 2008 (J. Harland and P. Manyem, eds.), Australian Computer Society, pp. 125-131, 2008.
5. D. V. Bailey, B. Baldwin, L. Batina, D. J. Bernstein, P. Birkner, J. W. Bos, G. V. Damme, G. Meulenaer, J. Fan, T. Güneysu, F. Gurkaynak, T. Kleinjung, T. Lange, N. Mentens, C. Paar, F. Regazzoni, P. Schwabe, L. Uhsadel, "The Certicom Challenges ECC2-X", Cryptology ePrint Archive, Report 2009/466, 2009.
6. D. V. Bailey, L. Batina, D. J. Bernstein, P. Birkner, J. W. Bos, H. Chen, C. Cheng, G. V. Damme, G. Meulenaer, L. J. D. Perez, J. Fan, T. Guneysu, F. Gurkaynak, T. Kleinjung, T. Lange, N. Mentens, R. Niederhagen, C. Paar, F. Regazzoni, P. Schwabe, L. Uhsadel, A. V. Herrewege, and B. Yang, "Breaking ECC2K-130", Cryptology ePrint Archive, Report 2009/541, 2009.
7. D. J. Bernstein, "Batch binary Edwards", In Crypto 2009, volume 5677 of LNCS, pages 317-336, 2009.

8. D. J. Bernstein, T. Lange, and P. Schwabe, “On the correct use of the negation map in the Pollard rho method”, PKC 2011 (D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, eds.), LNCS, vol. 6571, Springer, 2011.
9. A. V. Bessalov, “A method of solution of the problem of taking the discrete logarithm on an elliptic curve by division of points by two”, Cybermetics and Systems Analysis, Vol.37, No.6, pp.820-823, 2001.
10. J. W. Bos, T. Kleinjung, and A. K. Lenstra, “On the use of the negation map in the Pollard Rho method”, ANTS IX (G. Hanrot, F. Morain, and E. Thomé, eds.), LNCS, vol. 6197, Springer, pp. 66-82, 2010.
11. R. P. Brent and J. M. Pollard, “Factorization of the eighth Fermat number”, Mathematics of Computation, 36: pp. 627-630, 1981.
12. H. Cohen, “A Course in Computational Algebraic Number Theory”, volume 138 of Graduate Texts in Mathematics. Springer-Verlag, 1993.
13. W. Diffie and M. Hellman, “New Directions in cryptography”, IEEE Transactions on Information Theory, volume 22, pp. 644-654, 1976.
14. FIPS 186-2, “Digital signature standard”, Federal Information Processing Standards Publication 186-2, February 2000.
15. K. Fong, D. Hankerson, J. Lopez, A. Menezes, “Field Inversion and Point Halving Revisited”, IEEE Trans. Computers 53(8): 1047-1059, 2004.
16. R. Gallant, R. Lambert and S. Vanstone, “Improving the parallelized Pollard lambda search on binary anomalous curves”, Mathematics of Computation, Volume 69, pp. 1699-1705, 1999.
17. D. Hankerson, J. Lopez-Hernandez, and A. Menezes, “Software Implementatin of Elliptic Curve Cryprography over Binary Fields”. In: Proceedings of CHES 2000. LNCS 1965, pp. 1-24. Springer, 2001.
18. R. Harley, Elliptic curve discrete logarithms project, Avaliable from http://pauillac.inria.fr/∼harley/ecdl/.
19. B. Harris, “Probability Distribution Related to Random Mappings”, Ann. Math. Statist. 31, pp. 1045-1062, 1960.
20. E. Knudsen, “Elliptic scalar multiplication using point halving”, Advances in Cryptology-ASIACRYPT’99, Lecture Notes in Computer Science 1716:135C149, 1999.
21. D. E. Knuth, “The Art of Computer Programming”, Vol. 3, 2nd ed, Addison-Wesley, Reading, Mass, 1981.
22. N. Koblitz, “Elliptic curve cryptosystems”, Mathematics of Computation, 48, pp. 203-209, 1987.
23. V. Miller, “Use of elliptic curves in cryptography”, Advances in Cryptology: proceedings of Crypto’85, LNCS 218, pp. 417-426, New York: Springer-Verlag, 1986.
24. P. L. Montgomery, “Speeding the Pollard and elliptic curve methods of factorization”, Mathematics of Computation 48, pp. 243-264, 1987.
25. National Institute for Standards and Technology, “Digital signature standard”, Federal information processing standard, U.S. Department of Commerce, FIPS PUB 186, Washington, DC, 1994.
26. J. M. Pollard, “A Monte Carlo method for factorization”, BIT 15, no. 3, pp. 331-335, 1975.
27. J. M. Pollard, “Monte Carlo methods for index computation mod p”, Mathematics of Computation, 32, pp. 918-924, 1978.
28. J. Sattler and C. P. Schnorr, “Generating random walks in groups”, Ann. Univ. Sci. Budapest. Sect. Comput., 6:65-79, 1985.
29. C. P. Schnorr, H. W. Lenstra, “A Monte Carlo Factoring Algorithm with Linear Storage”, Math. Comp. 43(167), pp. 289-311, 1984.

30. R. Schroeppel, “Elliptic curve point halving wins big”, 2nd Midwest Arithmetical Geometry in Cryptography Workshop, Urbana, Illinois, November 2000.
31. R. Schroeppel, “Elliptic curve point ambiguity resolution apparatus and method”, International Application Number PCT/US00/31014, filed 9 November 2000, publication number WO 01/35573 A1, 17 May 2001.
32. E. Teske, “Speeding up Pollard’s rho method for computing discrete logarithms”, in Algorithmic Number Theory Symposium (ANTS IV), LNCS 1423, Springer-Verlag, pp. 541-553, 1998.
33. E. Teske, “On random walks for Pollard’s rho method”, Mathematics of Computation 70(234), pp. 809-825, 2001.
34. P. van Oorschot and M. Wiener, “Parallel collision search with cryptanalytic applications”, Journal of Cryptology, 12, pp. 1-28, 1999.
35. M. Wiener and R. Zuccherato, “Faster attacks on elliptic curve cryptosystems”, Selected Areas in Cryptography’98, LNCS 1556, pp. 190-120, Springer-Verlag, 1998.