Computing Elliptic Curve Discrete Logarithms with the Negation Map [⋆]

Ping Wang and Fangguo Zhang

School of Information Science and Technology,
Sun Yat-Sen University, Guangzhou 510275, China
isszhfg@mail.sysu.edu.cn

Abstract. It is clear that the negation map can be used to speed up the computation of elliptic curve discrete logarithms with the Pollard rho method. However, the random walks defined on elliptic curve points equivalence class $\{\pm P\}$ used by Pollard rho will always get trapped in fruitless cycles. We propose an efficient alternative approach to resolve fruitless cycles. Besides the theoretical analysis, we also examine the performance of the new algorithm in experiments with elliptic curve groups. The experiment results show that we can achieve the speedup by a factor extremely close to $\sqrt{2}$, which is the best performance one can achieve in theory, using the new algorithm with the negation map.

Keywords: Pollard rho method, negation map, fruitless cycles, elliptic curve discrete logarithm, random walk.

1 Introduction

The discrete logarithm problem(DLP) is an important problem in modern cryptography. The security of various cryptographic systems and protocols relies on the presumed computational difficulty of solving the discrete logarithm problem, such as the ElGamal signature and encryption schemes [7], the U.S. governments Digital Signature Algorithm(DSA) [8], the Schnorr signature scheme [18], etc. Originally, they worked with multiplicative groups of finite prime fields. Once elliptic curve cryptosystems were proposed by Koblitz [13] and Miller [15], analogous practical systems based on the DLP in groups of points of elliptic curves over finite fields were designed [14].

For elliptic curve discrete logarithm problem(ECDLP), Pollard rho method [17] and its parallelized variant [21] are at present known as the best generic algorithms. These methods work for any cyclic groups and do not make use of any additional structure present in elliptic curve groups. By exploiting the group structure of elliptic curves over finite fields, Gallant, Lambert and Vanstone [9], and Wiener and Zuccherato [22] pointed out that Pollard rho method can be speedup by defining the random walk on the equivalence classes.

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

For point P of elliptic curves over finite fields $GF(p)$ or binary extension fields $GF(2^m)$, it is trivial to determine its inverse $-P$. Therefore, we can consider the negation map as the equivalence relation $\sim$, and define the random walk on the equivalence classes to halve the search space. Theoretically, one can achieve a speedup by a factor of $\sqrt{2}$ when solving ECDLP. Furthermore, for the elliptic curves over $GF(2^m)$ with coefficients in $GF(2)$, called anomalous binary curves or Koblitz curves, it is suggested to define the equivalence relation by combining the Frobenius endomorphism and the negation map to get a speedup of $\sqrt{2m}$ [9] [22]. For the Jacobian group of hyperelliptic curves over finite fields, one can obtain a speedup of $\sqrt{t}$ if there is an automorphism of order t [6].

However, when computing ECDLP with the Pollard rho method, one define random walks on the equivalence classes to make use of the negation map, which always leads to fruitless cycles. Several works [3] [6] [9] [22] have analyzed the appearance of fruitless cycles, and various methods have been proposed to deal with them. Methods from previous works can be summarized into the following aspects: 1) look ahead technique to reduce the short fruitless cycles, 2) iteration functions with doubling to reduce the occurrence and recurrence of the cycles, 3) cycle detection approaches and 4) cycle escape techniques. In practice, one may combine these methods to achieve the best performance.

The purpose of using the negation map is to obtain a speedup, hopefully by a factor of $\sqrt{2}$. However, currently the general result reported in the literature is the speedup by a factor of 1.29 [3], except for the case of combining the Frobenius endomorphism and the negation map for the Koblitz curves. We analyze the previously published methods to deal with the fruitless cycles, and show their inefficiency. This may explain why it is seldom to make use of the negation map to solve ECDLPs in practice, such as Certicom ECC challenges, except for the cases of Koblitz curves. We note that similar efforts to achieve the speedup by a factor close to $\sqrt{2}$ for SIMD (single instruction, multiple data) architectures with similar strategy but different approach has been independently done by Bernstein, Lange and Schwabe [2].

We describe effective alternative approaches based on cycle detection and escape strategy to resolve fruitless cycles. More precisely, we analyze the precise probabilities for appearance of the fruitless $2t$-cycles, and discuss the previous methods, then propose alternative iteration functions for the rho method based on an efficient cycle detection and escape method. Besides the theoretical analysis, we also compare their performances in experiments with elliptic curve groups. The experiment results show that we can achieve the speedup by a factor extremely close to $\sqrt{2}$, which is the best performance one can achieve in theory, using the new algorithm with the negation map.

The remainder of this paper is organized as follows. In Section 2, we recall the Pollard rho method for elliptical curve discrete logarithm computation and fruitless cycles, and also analyze the previous methods to handle these cycles. We propose alternative iteration functions for the rho method based on an efficient fruitless cycles detection method in section 3 and present our experiments in section 4. We conclude the paper in section 5.

2 Preliminaries

In this section, we describe the Pollard rho method for elliptical curve discrete logarithm computation and fruitless cycles, and then discuss the previous methods to handle these cycles.

Let E be an elliptic curve defined over a finite field 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, $0 \leq k < n$, called the logarithm of Q to the base P, denoted $\log_P Q$. The problem of finding k, given P, Q, and the parameters of E, is known as the *elliptic curve discrete logarithm problem*(ECDLP). In elliptic curve cryptography, the major security consideration is the intractability of the ECDLP.

2.1 Pollard rho Method

Pollard [17] 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 [5]. 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 $i \neq j$. Under the assumption that an iteration function $F : G \to G$ behaves like a truly random mapping and the initial value X_0 is a randomly chosen group element, the expected number of evaluations before a match appears is $E(\mu + \lambda) = \sqrt{\pi|G|/2}$ [11].

Pollard rho method 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. Here, we describe Teske's r-adding walk [19], a modified version of the rho method to compute ECDLPs.

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 $Q = kP$, we divide the group G into r subsets: $S_1, S_2, \cdots, S_r$ of roughly equal size, and the r-adding walk define the iteration function $F : G \to G$ as follows:

$$Y_{i+1} = F(Y_i) = Y_i + (m_j P + n_j Q) \quad \text{for } Y_i \in S_j \text{ and } j \in [1, r]$$

where m_j and n_j randomly chosen from $[0, n-1]$ and $(m_j P + n_j Q)$ can be precomputed for $j = 1, 2, \cdots, r$. 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 sequences of (a_i) and (b_i) can be updated as follows,

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

Hence, as soon as we have a match (Y_i, Y_j), we have the following equation:

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

Since $Q = kP$, this gives

$$a_i + b_i k \equiv (a_j + b_j k) \bmod n$$

Now, if $\gcd(b_i - b_j, n) = 1$, we get $k = (a_j - a_i)(b_i - b_j)^{-1} \bmod n$. Due to the method of Pohlig and Hellman [16], in practice applications the group order n is prime, so that it is very unlikely that $\gcd(b_i - b_j, n) > 1$ if n is large. Teske [19] found experimentally that r-adding walk with $r \geq 20$ perform very close to a random walk.

2.2 Negation Map and Fruitless Cycles

Notice that given a point $P = (x, y)$ on an elliptic curve over finite field, it is trivial to determine its negative. That is $-P = (x, -y)$ for the elliptic curve E over $GF(p)$ where p is an odd prime, and $-P = (x, x + y)$ for the elliptic curve E over $GF(2^m)$. Thus, at every step of the iteration for the Pollard rho method, both Y_i and $-Y_i$ could be easily computed.

Therefore, we can consider the negation map as the equivalence relation $\sim$, and define the random walk of the rho method on the equivalence classes to reduce the size of the space that is being searched by a factor of 2. We can do this by replacing Y_i with $\pm Y_i$ at each step in a canonical way. A simple way to do this is to choose the one that has smallest y coordinate when its binary representation is interpreted as an integer. Because the extra computational effort in determining which of Y_i and $-Y_i$ to accept is negligible, theoretically, the expected running time of the algorithm will be reduced by a factor of $\sqrt{2}$. This improvement is valid for any elliptic curve.

Fruitless Cycles. However, define the iteration function on such equivalence classes always leads to trivial fruitless cycles. Suppose that $Y_i \in S_j$ and $Y_{i+1} = -(Y_i + m_j P + n_j Q)$ (with probability $\frac{1}{2}$, for some j), if $Y_{i+1} \in S_j$ (with probability $\frac{1}{r}$), then $Y_{i+2} = -(Y_{i+1} + m_j P + n_j Q) = Y_i$ (with probability 1), thereby the random walk falls into a useless 2-cycle. It follows that a fruitless 2-cycle starts from a random point with probability $\frac{1}{2r}$ [6]. Let $R_j = m_j P + n_j Q$, then we can denote this 2-cycle as follows,

$$P_i \to -(P_i + R_j) \to P_i.$$

Similarly, the random walk may also fall into fruitless 4-cycle, 6-cycle, 8-cycle and so on with different probabilities. Typically, a fruitless 4-cycle without proper sub-cycle may have the form,

$$P_i \to P_i + R_j \to -(P_i + R_j + R_l) \to -(P_i + R_l) \to P_i.$$

The cycle may be entered at any of its four point. We will discuss in detail the probabilities for a random point falls into these fruitless cycles in next section.

Using Frobenius Endomorphism. Notice that for Koblitz curves, the map $\phi : E(F_{2^m}) \to E(F_{2^m})$ 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 by combining the Frobenius endomorphism and the negation map to get a speedup of $\sqrt{2m}$ [9] [22].

Based on Harley's work on ECC2K-95/ECC-2K108 [10] and [9], Bailey *et al.* [1] proposed an efficient and practical alternative iteration function for the rho method, which can successfully avoid the fruitless cycles. The new iteration function is given by

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

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, [1] 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.

Obviously, such variant iteration function has unique properties. For some integer i, j, t, we have

$$\text{if } Y_i = \phi^t(Y_j), \text{ then } Y_{i+1} = \phi^t(Y_{j+1});$$

$$\text{if } Y_i = -Y_j, \text{ then } Y_{i+1} = -Y_{j+1}.$$

That is the variant iteration function is well defined on equivalence classes, and can combine with the distinguished points technique to achieve a speedup of $\sqrt{2m}$. However, this method can only work with Koblitz curves.

Dealing with Fruitless Cycles. To reduce the occurrence of 2-cycles, Wiener and Zuccherato [22] propose to use a look-ahead technique which proceeds as follows. For each step of the random walk, check the current point and the coming next point, if both of them belong to the the same partition, then replace the next point by a new point with certain deterministic rules, thus reduce the probability that two successive points belong to the same partition. More precisely, for $Y_i \in S_j$ define the next point as follows,

$$Y_{i+1} = \begin{cases} \sim (Y_i + R_j) & \text{if } Y_{i+1} \notin S_j \quad (1) \\ \sim (Y_i + R_{(j+t) \bmod r}) & \text{if } Y_{i+1} \notin S_{(j+t) \bmod r} \text{ for the minimal} \\ & t \text{ in } [1, r-1] \quad (2) \\ \text{a new random point} & \text{otherwise} \quad (3) \end{cases}$$

The third case is a very low probability event, expected to happen once every r^r steps. The expected cost per step of the walk is increased by a factor, which lies between $1 + \frac{1}{r}$ and $1 + \frac{1}{r-1}$.

Although the look-ahead technique reduces the frequency of 2-cycles, they may still occur [22]. Furthermore, Bos *et al.* [3] pointed out that this 2-cycles reduction technique introduces new 2-cycles and 4-cycles, which occurs with probability $\frac{1}{2r^3}$ and $\frac{r-1}{4r^3}$, respectively. Hence, they suggested the following alternative iteration function to avoid recurring fruitless cycles. If $Y_i \in S_j$, then

$$Y_{i+1} = \begin{cases} \sim (Y_i + R_j) & \text{if } Y_{i+1} \notin S_j \quad (1) \\ \sim 2Y_i & \text{otherwise} \quad (2) \end{cases}$$

However, it is well known that just addressing 2-cycles does not solve the problem of fruitless cycles, because longer cycles will occur as well. Reducing their occurrence requires additional overhead on top of what is already incurred to reduce 2-cycles. Given that fruitless cycles are unavoidable, they must be effectively dealt with when they occur.

Gallant *et al.* [9] proposed a general approach to detect cycles and to escape from them: after α steps record a length β sequence of successive points and compare the next point to these β points. If a cycle is detected a cycle representative point Y_i' is chosen deterministically. We named this method as $\alpha\beta$-cycle detection method. There are several possible ways to escape the cycle from Y_i' by a modified iteration. One may set $Y_{i+1} = Y_i' + R'$ where R' is a distinct precomputed value that does not depend on the escape-point, or one may set $Y_{i+1} = Y_i' + R_j'$ where $Y_i' \in S_j$ and R_j' from a distinct list of r precomputed values $R_1', R_2', \ldots, R_r'$. Bos *et al.* [3] analyze these cycle escape methods, and suggested that it is better to set $Y_{i+1} = 2Y_i'$.

Because 2-cycles are more frequent, in practice one may combine the 2-cycle reduction method with the $\alpha\beta$-cycle detection and escape method to deal with fruitless cycles.

3 New Alternative Approaches

In this section we discuss fruitless cycles in greater detail and analyze the previous methods. Then we propose alternative iteration functions for the rho method based on an efficient cycle detection and escape method.

3.1 Motivation

To better understand the issue of fruitless cycles, we need to know the exact probabilities for a random point falls into fruitless $2t$-cycles, where t equal to $1, 2, 3, \dots$, and so on. More precisely, we have the following fact.

Theorem 1. *When computing discrete logarithms with Pollard rho method, define the r-adding walk on equivalence of the negation map, then the probability for a random point falls into fruitless* $2t$*-cycle is* $c\frac{t!(r-1)!}{2^t r^{2t-1}(r-t)!}$*, where* $c \in [\frac{1}{2}, 1]$ *is a constant for each positive integer* t*.*

Proof. It is clear that the probability for a random point falls into a fruitless $2t$-cycle without considering excluding certain sub-cycles is

$$\begin{aligned} Pr(2t\text{-cycle with sub-cycles}) &= \frac{r-1}{r} \cdot \frac{r-2}{r} \cdots \frac{r-(t-1)}{r} \cdot \frac{t}{2r} \cdot \frac{t-1}{2r} \cdots \frac{1}{2r} \\ &= \frac{t!(r-1)!}{2^t r^{2t-1}(r-t)!} \end{aligned}$$

However, the above probability for a random point falls into a fruitless $2t$-cycle include the cases that the point falls into certain sub-cycles with the cycle length less than $2t$. The sub-cycles accounts for a certain proportion of the total cases, thus

$$Pr(2t\text{-cycle}) = c \cdot Pr(2t\text{-cycle with sub-cycles}), \text{ where } c \in (0, 1].$$

For the trivial case $t = 1$, there is no sub-cycle, then $c = 1$.

For the cases $t = 2$ and 3, the sub-cycles accounts for half of the total cases, that is $c = \frac{1}{2}$.

For the rest of cases $t \geq 4$, there is always sub-cycles, hence $c < 1$. For each $t \geq 4$, let N_2, N_4, N_6, …, $N_{2(t-1)}$ be the numbers of sub-2-cycles, sub-4-cycles, sub-6-cycles, …, sub-2$(t-1)$-cycles, and let A_n^m denote $\frac{n!}{(n-m)!}$. Then we have

$$\begin{aligned} N_2 &= A_{t-1}^{t-2} \\ N_4 &= A_{t-2}^{t-3}(A_2^1 - A_1^0) \triangleq A_{t-2}^{t-3} M_4 \\ N_6 &= A_{t-3}^{t-4}(A_3^2 - A_2^1 - A_1^0 M_4) \triangleq A_{t-3}^{t-4} M_6 \\ N_8 &= A_{t-4}^{t-5}(A_4^3 - A_3^2 - A_2^1 M_4 - A_1^0 M_6) \triangleq A_{t-4}^{t-5} M_8 \\ &\cdots \\ N_{2(t-1)} &= A_1^0(A_{t-1}^{t-2} - A_{t-2}^{t-3} - A_{t-3}^{t-4} M_4 - \ldots - A_2^1 M_{2(t-3)} - A_1^0 M_{2(t-2)}) \end{aligned}$$

It is easy to check that, for $t \geq 4$

$$\sum_{i=1}^{t-1} N_{2i} < \frac{A_t^{t-1}}{2}$$

thus, we have

$$\begin{aligned} c &= \frac{A_t^{t-1} - \sum_{i=1}^{t-1} N_{2i}}{A_t^{t-1}} \\ &> \frac{1}{2} \end{aligned}$$

Combine all the above cases, we have

$$Pr(2t\text{-cycle}) = c\frac{t!(r-1)!}{2^t r^{2t-1}(r-t)!}, \text{ where } c \in [\frac{1}{2}, 1].$$

$\square$

According to Theorem 1, we present the precise probabilities for a random point falls into fruitless $2t$-cycles in Table 1, where t can be 1, 2, 3 and so on, and r equals to 20, 128 and 1024, the typical values suggested by the literature.

Table 1. Probabilities to fall into fruitless $2t$-cycles with different r.

t	c	Pr($2t$-cycles)	$r = 20$	$r = 128$	$r = 1024$
1	1	$\frac{1}{2r}$	$2^{-5.3}$	$2^{-8.0}$	$2^{-11.0}$
2	$\frac{1}{2}$	$\frac{r-1}{4r^3}$	$2^{-10.7}$	$2^{-16.0}$	$2^{-22.0}$
3	$\frac{1}{2}$	$\frac{3(r-1)(r-2)}{8r^5}$	$2^{-14.6}$	$2^{-22.4}$	$2^{-31.4}$
4	$\frac{13}{24}$	$\frac{39(r-1)(r-2)(r-3)}{48r^7}$	$2^{-18.0}$	$2^{-28.4}$	$2^{-40.3}$
5	$\frac{71}{120}$	$\frac{71(r-1)(r-2)\cdots(r-4)}{32r^9}$	$2^{-21.2}$	$2^{-34.0}$	$2^{-48.9}$
6	$\frac{461}{720}$	$\frac{461(r-1)(r-2)\cdots(r-5)}{64r^{11}}$	$2^{-24.3}$	$2^{-39.3}$	$2^{-57.2}$
7	$\frac{3447}{5040}$	$\frac{3447(r-1)(r-2)\cdots(r-6)}{128r^{13}}$	$2^{-27.2}$	$2^{-44.5}$	$2^{-65.3}$
8	$\frac{29093}{40320}$	$\frac{29093(r-1)(r-2)\cdots(r-7)}{256r^{15}}$	$2^{-30.1}$	$2^{-49.5}$	$2^{-73.2}$
9	$\frac{39049}{51840}$	$\frac{273343(r-1)(r-2)\cdots(r-8)}{512r^{17}}$	$2^{-32.9}$	$2^{-54.4}$	$2^{-81.0}$
10	$\frac{113173}{145152}$	$\frac{2829325(r-1)(r-2)\cdots(r-9)}{1024r^{19}}$	$2^{-35.7}$	$2^{-59.1}$	$2^{-88.6}$
11	$\frac{10666301}{13305600}$	$\frac{31998903(r-1)(r-2)\cdots(r-10)}{2048r^{21}}$	$2^{-38.5}$	$2^{-63.7}$	$2^{-96.1}$
12	$\frac{392743957}{479001600}$	$\frac{392743957(r-1)(r-2)\cdots(r-11)}{4096r^{23}}$	$2^{-41.4}$	$2^{-68.2}$	$2^{-103.5}$
13	$\frac{1040212291}{1245404160}$	$\frac{5201061455(r-1)(r-2)\cdots(r-12)}{8192r^{25}}$	$2^{-43.3}$	$2^{-71.6}$	$2^{-109.8}$
14	$\frac{73943424413}{87178291200}$	$\frac{73943424413(r-1)(r-2)\cdots(r-13)}{16384r^{27}}$	$2^{-47.3}$	$2^{-77.0}$	$2^{-118.0}$
15	$\frac{8028493013}{9340531200}$	$\frac{280997255455(r-1)(r-2)\cdots(r-14)}{8192r^{29}}$	$2^{-50.5}$	$2^{-81.2}$	$2^{-125.1}$

Notice that the probability to enter a fruitless cycle decrease with increasing r. However, it [3] shows that increasing r-values always leads to poor performance for the iteration functions. Generally, r cannot be chosen large enough to both avoid fruitless cycles and achieve adequate performance. Therefore, we need certain techniques to deal with fruitless cycles.

As we mentioned above, currently the general solution is combining the 2-cycle reduction method with the $\alpha\beta$-cycle detection and doubling-based escape to achieve a speedup by a factor of 1.29 using the negation map, which is less than the speedup factor of $\sqrt{2}$ in theory. Bos *et al.* [3] concluded that "We measured to what extent our failure to achieve a speedup by a factor of $\sqrt{2}$ can be blamed on cycle detection and escape and other overheads". Therefore, we are expected to provide a better cycle detection method.

3.2 The Main Algorithm

To solve the fruitless cycles, our basic idea is quite simple. The random walk proceeds as usual, whenever it falls into a fruitless cycle, it can escape the cycle from certain escape-point. If $Y_i \in S_j$, then our iteration function can be defined as follows,

$$Y_{i+1} = \begin{cases} \sim 2Y_i & \text{if } Y_i \text{ is an escape point} \quad (1) \\ \sim (Y_i + R_j) & \text{otherwise} \quad (2) \end{cases}$$

To make the above iteration function efficient, one needs an effective method to detect the fruitless cycles of various length with different probabilities and an efficient way to determine proper escape point. The main algorithm for the solution works as follows: We fix an integer N and use two auxiliary variables, say Y_{min} and Y_{new}, where Y_{min} is the minimum point of N successive points produced by the iteration function F, and Y_{new} is the newly computed minimum point of next N successive points, here the minimum point means the point has the minimum x-coordinate (or y-coordinate). If $Y_{new} = Y_{min}$, which means the random walk has fall into certain fruitless cycle, we can assign the minimum point as the escape point and escape the cycle by double Y_{min}. Otherwise, replace Y_{min} by Y_{new}, and continue to find the next Y_{new}. That is, during the random walk we keep the minimum point of N successive points, if the random walk entered certain fruitless cycle, we will get collision between the minimum points very soon. By properly set the integer N, one can detect and escape these fruitless cycles whose length less than or equal to N.

Generally, one can determine the parameter N according to Table 1, which means N varies with different r and the size of the group G. Typically, one may set $N = 12, 16, 20$. However, we will show later that when the random walk falls into fruitless cycles, on average it needs about $2.75N$ additional iterations before we detect it using the above algorithm. Hence it will be inefficient to detect the more frequent fruitless 2-cycles and 4-cycles, especially when r is small.

Therefore, it is necessary to improve the above algorithm to detect the short cycles more efficiently, and we illustrate as follows: For efficiency, generally we assume N is multiple of 4, here we demonstrate by setting $N = 16$. For instance, we want to find the minimum point Y_{new} from 16 successive points $\{Y_1, Y_2, \dots Y_{16}\}$. We proceed as follows, first we compare Y_1 and Y_2, Y_3 and Y_4, respectively. Let $Y_{1,2} = min(Y_1, Y_2)$, and $Y_{3,4} = min(Y_3, Y_4)$. Then we compare $Y_{1,2}$ and $Y_{3,4}$, if $Y_{1,2} = Y_{3,4}$, the random walk has entered fruitless cycle, we just set $Y_{1,2}$ be the escape point, otherwise let $Y_{1,4} = min(Y_{1,2}, Y_{3,4})$, which is the minimum point of Y_1, Y_2, Y_3 and Y_4.

Similarly, we then compare Y_5 and Y_6, Y_7 and Y_8, respectively. Let $Y_{5,6} = min(Y_5, Y_6)$, and $Y_{7,8} = min(Y_7, Y_8)$. If $Y_{5,6} = Y_{7,8}$, we set $Y_{5,6}$ be the escape point, otherwise let $Y_{5,8} = min(Y_{5,6}, Y_{7,8})$, which is the minimum point of Y_5, Y_6, Y_7 and Y_8. Then we compare $Y_{1,4}$ and $Y_{5,8}$, if $Y_{1,4} = Y_{5,8}$, the random walk has entered fruitless cycle, we set $Y_{1,4}$ be the escape point, otherwise let $Y_{1,8} = min(Y_{1,4}, Y_{5,8})$, which is the minimum point of $\{Y_1, Y_2, \dots Y_8\}$.

We show the above procedures in the following Fig. 1.

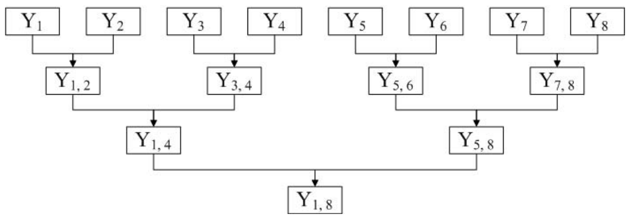

Fig. 1. Find the minimum point of $\{Y_1, Y_2, \dots Y_8\}$ and detect the fruitless 2-cycles or 4-cycles, simultaneously. For efficiency, when r is small, e.g. $r = 20$, it is necessary to check whether $Y_{1,4} = Y_{5,8}$ to detect the possible 4-cycles. However, one may ignore such check and just set $Y_{1,8} = min(Y_{1,4}, Y_{5,8})$ when r is big, e.g. $r = 1024$, where the possible 4-cycles are rare.

In the same way, we can get $Y_{9,16}$ the minimum point of $\{Y_9, Y_{10}, \dots Y_{16}\}$, and let $Y_{new} = min(Y_{1,8}, Y_{9,16})$, which is the minimum point of $\{Y_1, Y_2, \dots Y_{16}\}$. Now, if $Y_{new} = Y_{min}$, which means the random walk has fall into certain fruitless cycle, we set the minimum point Y_{min} be the escape point. Otherwise, replace Y_{min} by Y_{new}, and continue to find the next Y_{new}, which is the minimum point of next 16 successive points.

Remark 1. In fact to find the minimum one from two points, it is not necessary to find the one with exact minimum x-coordinate (or y-coordinate). For the new algorithm, one can define the minimum point as the point whose least 32 bits is the minimum when interpreted as an integer. Hence the cost of operation to choose the minimum one from two points is much cheaper than compare two points or find the one with exact minimum x-coordinate (or y-coordinate).

Obviously, when we set certain point Y_i as the minimum point or the escape point, we need track the corresponding indices a_i and b_i, such that $Y_i = a_iP + b_iQ$. Note that the above algorithm leads to the new iteration function we have defined in this section. More important, Y_{i+1} still depends solely on Y_i, which is a requirement for distinguished point method to work.

Here, we demonstrate the new iteration function with the parallelized Pollard rho, that is on input an initial point Y_0, the algorithm proceed with the new iteration function, and finally return a distinguished point. Assuming the function $F : G \to G$ is a traditional iteration function, that is, if $Y_i \in S_j$,

$F(Y_i) =\sim (Y_i + R_j)$. Let DP be the set of distinguished points and EP be a variable to keep escape point. Then we have the following Algorithm 1 for the typical case $N = 16$.

Algorithm 1 New Algorithm for Pollard rho with the Negation Map

```
Input: Initial value $Y_0$
Output: A distinguished point $\in DP$
1: $Y_{min} \leftarrow 0$, $Y_{new} \leftarrow 0$, $EP \leftarrow 0$
2: while True do
3:     if $EP \neq 0$ then
4:         $Y_0 \leftarrow 2EP$, $EP \leftarrow 0$
5:     end if
6:     if $Y_0 \in DP$ then
7:         return $Y_0$
8:     end if
9:     for $i = 1$ to 4 do
10:        for $j = 1$ to 4 do
11:            $Y_j \leftarrow F(Y_{j-1})$
12:            if $Y_j \in DP$ then
13:                return $Y_j$
14:            end if
15:        end for
16:        $Y1_i \leftarrow min(Y_1, Y_2)$, $Y2_i \leftarrow min(Y_3, Y_4)$
17:        if $Y1_i = Y2_i$ then
18:            $EP \leftarrow Y1_i$
19:            break
20:        else
21:            $Y3_i \leftarrow min(Y1_i, Y2_i)$
22:        end if
23:        if $i = 2$ then
24:            $Y4_1 \leftarrow min(Y3_1, Y3_2)$
25:        end if
26:    end for
27:    $Y4_2 \leftarrow min(Y3_3, Y3_4)$, $Y_{new} \leftarrow min(Y4_1, Y4_2)$
28:    if $Y_{new} = Y_{min}$ then
29:        $EP \leftarrow Y_{min}$
30:        break
31:    else
32:        $Y_0 \leftarrow Y_4$
33:    end if
34: end while
```

In fact, when we detect a cycle according to the new algorithm, we need to check whether it is a valid cycle, which figures out the ECDLP, or an invalid cycle. However, the probability that such cycle with length less than or equal to N is valid can be neglected. Therefore, for efficiency we just omit such check.

3.3 Analysis

For the new algorithm, there is a probability that we can not detect the collision or cycle immediately when it happens. On average it needs to compute 3 additional points to detect a fruitless 2-cycle and about 7 additional points to detect a fruitless 4-cycle when the random walk reachs certain cycle.

For the rest fruitless $2t$-cycles with much smaller probabilities, where $4 < 2t \leq N$, the new algorithm detect them by checking whether $Y_{new} = Y_{min}$. Generally with high probability, the new algorithm will detect such cycles within certain additional points. More precisely, we have the following theorem.

Theorem 2. *Under the assumption that the iteration function $F : G \to G$ is a random mapping, one detect fruitless $2t$-cycles with $4 \leq 2t \leq N$ using Algorithm 1. When the random walk reach certain fruitless cycle, the number of additional iterations before finding and escaping the cycle is $(k + \frac{1}{2})N$ with probability $1 - \frac{1}{2}(\frac{2}{3})^{(k-1)}$ where $k = 0, 1, 2, \cdots$. That is, the expected number of additional iterations is $2.75N$.*

Proof. Let I_i be the set that consists of the ith N successive values generated by iteration function F. That is,

$$I_i = \{Y_j \mid iN \leq j \leq (i+1)N - 1,\ j \geq 0\}, \text{ for } i = 0, 1, 2, \cdots, \lceil \frac{|G|}{N} \rceil - 1.$$

For finite group G, the sequence produced by F is eventually period. That is, for any fixed F and Y_0, there exist certain integers m and n, such that

$$I_m \bigcap I_n \neq \varnothing,\ m < n.$$

and

$$I_i \bigcap I_j = \varnothing, \text{ for } 0 \leq i, j < n \text{ and } i \neq j.$$

and also

$$I_{m+i} \bigcap I_{n+i} \neq \varnothing, \text{ for } i \geq 0.$$

To prove the theorem, we divide it into two cases, that is $k = 0$ and $k = 1, 2, \cdots$. For $k = 0$, let min_m and min_n be the minimum value of I_m and I_n, then

$$\begin{aligned} Pr(min_m = min_n) &= \frac{1}{N} \sum_{i=0}^{N-1} (\frac{1}{N} + \frac{2}{N} + \cdots + \frac{N-i}{N}) \frac{1}{N-i} \\ &= \frac{N^2 + 3N + 3}{4N^2} \\ &\approx \frac{1}{4} \end{aligned}$$

That is, the probability of successfully detecting the collision within the first two intersection sets is $\frac{1}{4}$.

For each of $k = 1, 2, \cdots$, we notice that, for two (intersected) sets I_i and I_j, let min_i and min_j be the minimum value of I_i and I_j, respectively, we have

$Pr(min_i = min_j) = |I_i \bigcap I_j|^2/N^2,$ where $|I_i \bigcap I_j|$ denotes the cardinality of $I_i \bigcap I_j$.

Therefore, we have

$$\begin{aligned} Pr(min_{m+k} = min_{n+k}) &= \frac{1}{N}(\frac{1}{N^2} + \frac{2^2}{N^2} + \cdots + \frac{(N-1)^2}{N^2}) \\ &= \frac{2N^2 - 3N + 1}{6N^2} \\ &\approx \frac{1}{3} \end{aligned}$$

Combining the above two cases, under the assumption that the iteration function $F : G \to G$ is a random mapping, one detect fruitless $2t$-cycles with $4 \le 2t \le N$ using Algorithm 1, the number of additional iterations before finding and escaping the cycle from the minimum values is $(k + \frac{1}{2})N$ with probability $1 - \frac{1}{2}(\frac{2}{3})^{(k-1)}$ where $k = 0, 1, 2, \cdots$.

Hence, the expected number of additional iterations is

$$(1 - \frac{1}{2}(\frac{2}{3})^{-1})\frac{1}{2}N + \sum_{k=1}^{\infty}((1 - \frac{1}{2}(\frac{2}{3})^{k-1}) - (1 - \frac{1}{2}(\frac{2}{3})^{k-2}))(k + \frac{1}{2})N \approx 2.75N$$

□

Therefore, the new algorithm is a probabilistic algorithm. There is a probability that we can not detect the collision or cycle immediately when it happens. However, with high probability, the random walk will escape the cycle within certain additional iterations.

Compare to the traditional iteration function without negation map, on average the new algorithm introduce one integer comparison for each iteration to track certain minimum point. For each 2-cycle and 4-cycle, on average it needs 3 additional iterations and 7 additional iterations to detect and escape the cycle. For the rest $2t$-cycles, where $2t \le N$, it is expected to compute $2.75N$ additional iterations to detect and escape the cycle. As a result, we resolved the fruitless cycles with negligible additional operations, that is we can achieve the speedup by a factor extremely close to $\sqrt{2}$ using the new algorithm with the negation map.

4 Experiments

To evaluate the efficiency of the new algorithm, we implemented and compared the parallelized Pollard rho method with the negation map and without the negation map for the Certicom ECCp-131 challenge [4]. We also examined the performances of the new algorithm with different r and N. In this section, we describe these experiments and analysis the results.

For our experiments, we briefly introduce the elliptic curve groups over prime fields and the notation we use in the following. Let q be a prime and let F_q denote F_q denote the field $\mathbb{Z}_q$ of integers modulo q. Let $a, b \in F_q$ such that $4a^3+27b^2 \neq 0$. Then the elliptic curve F_q is defined through the equation

$$E_{a,b} : y^2 = x^3 + ax + b.$$

The set of all solutions $(x, y) \in F_q \times F_q$ of this equation, together with the element $\mathcal{O}$ called the "point at infinity", forms a finite abelian group which we denote by $E_{a,b}(F_q)$. Usually, this group is written additively. Let $P \in E_{a,b}(F_q)$ be a point of prime order n, let G denote the subgroup of E generated by P. Given $Q \in G$, determine the integer $0 \leq k < n$ such that $Q = kP$.

The parameters can be found online on Certicom's website [4]. For the iteration function, we use the Teske's r-adding walk, and set r be 20, 128 and 1024, respectively.

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

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

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

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

and

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

This method is originally from Knuth's multiplicative hash function [12] and suggested by Teske [20]. From the theory of multiplicative hash functions we know that among all numbers between 0 and 1, choosing A as a rational approximation of $\frac{\sqrt{5}-1}{2}$ with a sufficiently large precision leads to the most uniformly distributed hash values, even for non-random inputs.

The purpose of our experiments is to examine the impact of additional operations to resolve fruitless cycles on the performance of the rho method. In more detail, we evaluate the numbers of iterations until a distinguished point is found for the parallelized Pollard rho method with the negation map and without the negation map, respectively. Generally, we first define the distinguishing property, and compute certain number of distinguished points using the two different ways, and then count the mean value of the numbers of iterations until a distinguished point is found for each way.

More precisely, to break Certicom ECCp-131, one may define the distinguishing property as the Hamming weight of the x-coordinate of the point less than 32 or 33. However, to examine the performance of the new algorithm, we define the distinguishing property as the Hamming weight of the least 32 bits of the x-coordinate of the point less than 8. Each point has probability almost exactly $(\binom{32}{0}+\binom{32}{1}+\cdots+\binom{32}{7})/2^{32} \approx 2^{-9.8937}$ of being a distinguished point. That is, it is expected to take about 951.26 iterations to find a distinguished point. In our experiments, we collect 1 million distinguished points for each way and count the corresponding mean value of the numbers of iterations until a distinguished point is found.

For Pollard rho with the negation map, we set $r = 20$, 128 and 1024, and choose $N = 14$, 8 and 6, respectively. The experiment results are given in Table 2. It confirms our theoretic analysis that the additional iterations introduced by dealing with the fruitless cycles can be neglected.

Table 2. The mean value of the numbers of iterations for each million samples

r	Iterations without Negation Map	Iterations with Negation Map
20	951.17	951.66
128	949.94	952.37
1024	951.02	949.53

In the experiment, we also examined the performances of the new algorithm to detect 2-cycles and 4-cycles within N iterations, it shows that when r is big, e.g. $r = 1024$, it is not necessary to detect the fruitless 4-cycles within the N iterations.

5 Conclusion

In this paper, we examined the exact probabilities for a random point falls into fruitless $2t$-cycles when computing ECDLPs using Pollard rho method with the negation map. We proposed effective alternative approaches based on cycle detection and escape strategy to resolve fruitless cycles. In conclusion, we resolved the fruitless cycles with negligible additional operations, one integer comparison for each iteration to track certain minimum point, that is we can achieve the speedup by a factor extremely close to $\sqrt{2}$ using the new approach with the negation map.

References

1. 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.
2. D. J. Bernstein, T. Lange and P. Schwabe, "On the correct use of the negation map in the Pollard rho method", To appear in PKC2011, 2011.

3. J. W. Bos, T. Kleinjung, A. K. Lenstra, “On the Use of the Negation Map in the Pollard Rho Method”, In Algorithmic Number Theory (ANTS) 2010, volume 6197 of LNCS, pp. 67-83, 2010.
4. Certicom, Certicom ECC Challenge, http://www.certicom.com/pdfs/cert_ecc_challenge.pdf, 2009.
5. H. Cohen, “A Course in Computational Algebraic Number Theory”, volume 138 of Graduate Texts in Mathematics. Springer-Verlag, 1993.
6. I. Duursma, P. Gaudry and F. Morain, “Speeding up the discrete log computation on curves with automorphisms”, ASIACRYPT 1999, LNCS 1716, pp. 103-121, Springer, Heidelberg, 1999.
7. T. ElGamal, “A public-key cryptosystem and a signature scheme based on discrete logarithms”, IEEE Transactions on Information Theory, volume 31, pp. 469-472, 1985.
8. FIPS 186-2, “Digital signature standard”, Federal Information Processing Standards Publication 186-2, February 2000.
9. 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.
10. R. Harley, Elliptic curve discrete logarithms project, Avaliable from http://pauillac.inria.fr/∼harley/ecdl/.
11. B. Harris, “Probability Distribution Related to Random Mappings”, Ann. Math. Statist. 31, pp. 1045-1062, 1960.
12. D. E. Knuth, “The Art of Computer Programming”, Vol. 3, 2nd ed, Addison-Wesley, Reading, Mass, 1981.
13. N. Koblitz, “Elliptic curve cryptosystems”, Mathematics of Computation, 48, pp. 203-209, 1987.
14. A. Menezes, P. van Oorschot, and S. A. Vanstone, “Handbook of applied cryptography”, CRC Press, 1996.
15. 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.
16. S. C. Pohlig and M. E. Hellman, “An improved algorithm for computing logarithms over GF(p) and its cryptographic significance”, IEEE-Transactions on Information Theory 24, pp. 106-110, 1978.
17. J. Pollard, “Monte Carlo methods for index computation mod p”, Mathematics of Computation, 32, pp. 918-924, 1978.
18. C. P. Schnorr, “Efficient signature generation by smart cards”, Journal of Cryptology, volume 4, pp. 161-174, 1991.
19. 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.
20. E. Teske, “On random walks for Pollard’s rho method”, Mathematics of Computation 70(234), pp. 809-825, 2001.
21. P. van Oorschot and M. Wiener, “Parallel collision search with cryptanalytic applications”, Journal of Cryptology, 12, pp. 1-28, 1999.
22. M. Wiener and R. Zuccherato, “Faster attacks on elliptic curve cryptosystems”, Selected Areas in Cryptography’98, LNCS 1556, pp. 190-120, Springer-Verlag, 1998.