A New Iterating Function
in the Pollard Rho Method for Discrete Logarithms

Maurizio Laporta and Alberto Pizzirani

Dipartimento di Matematica e Applicazioni
Università degli Studi di Napoli "Federico II"
Complesso Universitario di Monte S.Angelo
Via Cinthia, 80126 Napoli, Italy

Copyright © 2014 Maurizio Laporta and Alberto Pizzirani. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract
First we give a very short introduction to some algorithms based on the Pollard rho method for computing discrete logarithms. Then, we briefly discuss a new iterating function defined on elliptic curve groups. In particular, we show that fruitless cycles generated by our function are rather long and occur with a small probability.

Mathematics Subject Classification: 11Y16, 65C05, 94A60

Keywords: Pollard rho method, discrete logarithm, fruitless cycles

1 Introduction: the classical rho method

Let us consider the discrete logarithm problem (DLP) to the base \( g \in G \), where \( G \) is a finite abelian group with an operation written multiplicatively:

\[
g^x = y.
\]

Note that such an integer \( x \) exists if and only if \( y \in \langle g \rangle \), where \( \langle g \rangle \) is the cyclic subgroup of \( G \) generated by \( g \). Namely, assuming that the order of \( \langle g \rangle \) is \( n \), one may restrict the search for the solution to the interval \( 0 \leq x \leq n - 1 \). In particular, given nonzero \( a, b \) modulo a prime number \( p \), the classical DLP
is to find an integer $k$ such that $a^k \equiv b \pmod{p}$, where any DL $k$ should be regarded as being defined mod $p-1$, or modulo a divisor $d$ of $p-1$ whenever $a^d \equiv_p 1$. Analogous properties hold in the multiplicative group $\mathbb{F}_q^*$ of a finite field $\mathbb{F}_q$. Finally, if the operation of $G$ is assigned additively, then the equation in (1) becomes $xg = y$. This is the case of an elliptic group $E(\mathbb{F}_q)$, where the so called discrete logarithm problem on elliptic curves (ECDLP) is stated as: for $P, Q \in E(\mathbb{F}_q)$ find an integer $x$ such that $Q = xP$. There are several cryptographic applications of the DLP, where it turns out that the security of the cryptosystems relies on the difficulty of solving such a problem in a suitable group. Trying all possible values of $x$ to solve (1) becomes impractical when the solution is an integer of a large size. Some less trivial techniques have been found to attack DLP of cryptographic interests, and all the best methods have expected running time $O(\sqrt{|G|})$, where $|G|$ denotes the order of $G$ (see [5], [11] and [12]). Among such square-root algorithms, a so called Monte Carlo method was introduced by John M. Pollard [8] in 1978, namely the rho algorithm for the discrete logarithm in $\mathbb{F}_p^*$. However, it was immediately clear that more in general the same method applies to (1), whenever one can assign numerical labels to the elements of $G$ and it is possible to perform effectively the group operations. Further, it has been observed that the space requirements of the algorithms based on the rho method are negligible with respect to the running time. Finally, a basic feature of the rho method is that DLP computations can be easily distributed to many processors.

At the core of the rho algorithms design there is a suitable choice of an iterating function $f : G \rightarrow G$ that generates a sequence $x_{i+1} = f(x_i)$ from an arbitrarily chosen $x_0 \in G$. Since $G$ is finite, it is plain that there exist $i_0 < j_0$ such that $x_{i_0} = x_{j_0}$. This immediately yields $x_{i_0 + 1} = f(x_{i_0}) = f(x_{j_0}) = x_{j_0 + 1}$, and more in general one gets the collisions (or matches) $x_{i_0 + s} = x_{j_0 + s}$ for all integers $s \geq 0$. Thus, the resulting sequence $\{x_i\}_{i \geq 0}$ is cyclic having a divisor of $j_0 - i_0$ (proper or not) as period\(^2\). Bearing in mind the ECDLP, let us show how in the Pollard original method one could exploit collisions in order to solve instances of the DLP in an additive group $G$ with respect to a base $P \in G$:

\[ \text{given } Q \in G, \text{ find an integer } k \text{ such that } kP = Q. \quad (2) \]

The group $G$ is partitioned into three disjoint subsets $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ of approximately the same size, such that none of the $\mathcal{T}_i$ is a subgroup of $G$ and in practice it can be efficiently determined which $\mathcal{T}_i$ contains any given $y \in G$. Assuming that integers $a_i, b_i$ ($i = 1, 2$) can be chosen at random, the Pollard iterating function $f_\mathcal{P} : G \rightarrow G$ is defined as

\[ f_\mathcal{P}(x) = \begin{cases} 
\text{rand} & \text{if } x \in \mathcal{T}_1 \\
\text{rand} & \text{if } x \in \mathcal{T}_2 \\
\text{rand} & \text{if } x \in \mathcal{T}_3 
\end{cases} \]

\(^1\)Somewhere we write $a \equiv_m b$ to mean $a \equiv b \pmod{m}$. Hereafter, $p$ denotes always a prime.

\(^2\)One might diagram such a sequence with the Greek letter $\rho$, whose tail indicates the precyclic part, while the cyclic part is represented by the oval of the letter.
A new iterating function in the Pollard rho method

6793

From a starting point \(a_0P + b_0Q\) the sequence generated by \(f_\gamma\) takes the form \(\{u_jP + v_jQ\}_{j \geq 0}\) where \(u_j, v_j \in \mathbb{Z}/n\mathbb{Z}\) with \(n := |G|\). More precisely,

\[
(u_{j+1}, v_{j+1}) = (u_j, v_j) + (a_i, b_i) = (u_j + a_i, v_j + b_i) \pmod{n}.
\]

Recalling (2), a collision \(u_{j_0}P + v_{j_0}Q = u_{i_0}P + v_{i_0}Q\) is equivalent to the congruence \(k \equiv (v_{j_0} - v_{i_0})^{-1}(u_{i_0} - u_{j_0}) \pmod{n/d}\), where \(d := \text{g.c.d.}(n, v_{j_0} - v_{i_0})\). In particular, if \(d\) is sufficiently small, then it is possible to verify all the admissible \(d\) choices for \(k = (v_{j_0} - v_{i_0})^{-1}(u_{i_0} - u_{j_0}) + mn/d\) with \(m = 0, \ldots, d-1\).

The case \(n\) prime is particularly sensitive because there are only two possibilities, \(d = 1\) or \(d = n\). The first one occurs with high probability \(1 - 1/n\) and corresponds to a golden collision, i.e. it gives a solution of (2). In the rare case \(d = n\) one has to start over the algorithm to look for a fruitful collision.

2 The need of new iterating functions

Any iterating sequence of the type \(x_{i+1} = f(x_i)\) is a simulation of a walk in \(G\), whose finiteness ensures that at some point such a walk turns into a loop. The occurrence of the first collision strictly depends on the iterating function \(f\). For example, for \(G = \mathbb{F}_p^*\) it has to do with the so called epact of \(p\) with respect to \(f : \mathbb{F}_p^* \to \mathbb{F}_p^*\), i.e. the largest index \(e := e(p)\) such that \(x_0, x_1 = f(x_0), x_2 = f(x_1), x_3 = f(x_2), \ldots, x_{e-1} = f(x_{e-2})\) are distinct. In particular, linear functions of the form \(ax + b\) and quadratic functions like \(x^2\) produce bad epacts (see [5], [8]). Somehow, a motivation for the partitioning \(G\) into the rules \(\mathcal{T}_i\) for the Pollard original function\(^3\) is to exploit such long epacts in order to generate walks that do not indulge too long in any \(\mathcal{T}_i\). Since which set \(\mathcal{T}_i\) an element is in has seemingly nothing to do with the group \(G\), one may think of the walk \(\{u_jP + v_jQ\}_{j \geq 0}\) as random. So it is possible to apply a well known argument in probability theory\(^4\) to see that \(\sqrt{\pi |G|/2} \approx 1.2533\sqrt{|G|}\) might be assumed to be the average number of generated items needed to have more than 50% chances for a collision in an idealized rho method. Nevertheless, supported by experimental evidences, Teske [10] observed that the average performance of the Pollard classical function on \(\mathbb{F}_p^*\) is worse than expected for a random mapping, namely collisions occur after an average number of \(1.37\sqrt{p - 1}\) iterations. Even worse is the scenario regarding the generalization of the Pollard function for arbitrary groups. In this sense, the walks generated by the Pollard function are in general not random enough, from whence the need of alternative iterating functions. Arguing heuristically, Teske showed

\(^3\)In the additive notation, the function \(f_\gamma\) is quadratic on \(\mathcal{T}_1\) and linear on \(\mathcal{T}_2\) and \(\mathcal{T}_3\).

\(^4\)This is the so called birthday paradox.
that remarkable improvements are gained by increasing the number of the partition sets of $G$, namely up to $r = 100$ rules instead of $r = 3$ as in the Pollard case. Indeed, with an appropriate choice of the parameters Teske’s new functions yield an average performance that is hardly distinguishable from the performance of a random mapping. In [9] the $r$-adding walks and the $r + q$-mixed walks are defined, where $r$ and $r + q$ denote the number of rules for the respective iterating functions. Precisely, the two functions are defined in an additive group $G$ as follows. Given positive integers $r, q$, let $M_1, \ldots, M_r \in G$ be randomly chosen and let $\mathcal{H} : G \rightarrow \{1, \ldots, r\}, \mathcal{K} : G \rightarrow \{1, \ldots, r + q\}$ be hash functions. The functions $f_{\mathcal{H}} : G \rightarrow G$ and $f_{\mathcal{K}} : G \rightarrow G$ are respectively $r$-adding and $r + q$-mixed if

$$f_{\mathcal{H}}(X) := X + M_{\mathcal{H}(X)}, \quad f_{\mathcal{K}}(X) := \begin{cases} X + M_{\mathcal{K}(X)} & \text{if } 1 \leq \mathcal{K}(X) \leq r \\ 2X & \text{if } \mathcal{K}(X) \geq r + 1 \end{cases}.$$

Note that the Pollard function $f_P$ is $2 + 1$-mixed. Further, with the DLP (2) in mind, each $M_i := a_iP + b_iQ$ is precomputed with random $a_i, b_i \in \mathbb{Z}/|G|\mathbb{Z}$.

Beside the aforementioned questions on the running time, there is also the space complexity problem. A trivial detection of the collisions would take around $\sqrt{|G|}$ storage. However, at the cost of a little more computation, the space complexity is strongly improved by the Floyd cycle-finding method (also known as the tortoise and the hare algorithm), where, beyond the usual sequence $\{x_i\}_{i \geq 0}$, it is required to generate also $\{x_{2i}\}_{i \geq 0}$ by a double application of the iterating function at each step (see [5]). A tradeoff comes out from storing only distinguished points of a certain subset of $G$, that should be easily testable to avoid an high overhead on computations spread across multiple computers. The risk to be taken into account is that a walk could enter into an useless loop of non distinguished points which will be never stored.

### 3 Iterating on equivalence classes

Further improvements for the rho method might come from a reduction of the search space. The most usual way is to iterate a walk through the classes defined by some equivalence relation in $G$. Since a quotient group has order of the type $|G|/m$, a collision between classes should be found in an ideal time $\sqrt{\pi|G|/(2m)}$. It is well known that automorphisms of $G$ define equivalence relations in $G$. Except the trivial case of the identity map, the most handy automorphism is the negation map, $X \in G \rightarrow -X \in G$, whose order\footnote{It is tacitly understood that extreme cases, like groups of exponent 2, are not under discussion here because they are of no interest for DLP.} is 2. Indeed, pairing $X \in G$ with its inverse $-X$ halves the search space of the algorithm. However, dealing with equivalence classes, the most obvious usage of the automorphisms in the context of the Pollard rho method leads to useless
cycles trapping the random walks. In other words, the iterating process falls into a loop that keeps on generating always the same classes. Commonly called fruitless cycles, such loops might seriously affect the speed-up of the algorithm. In [6] it is showed how fruitless cycles of length 2 could easily appear when the rho algorithm is implemented with a r-adding function on G via the negation map. The same paper includes an analysis of the probability of t-cycles when a r-adding function is iterated via a general automorphism of order m. It turns out that such a probability reduces as r and t increase. On the other side, large values of r are impractical, while the most promising medium values are not always compatible with all environments (see [3]). So the detection of the cycles occurrence and their treatment are key topics in the development of the rho algorithms when dealing with automorphisms. In the literature there are many concrete proposals on the use of the negation map in the rho method. Mostly, they are a combination of several ideas for avoiding, detecting, and escaping fruitless cycles. A good review of such proposals is the Appendix of [1], where it is remarked that none of the proposed algorithms perform efficiently on SIMD architectures because of frequent conditional operations. In the same [1] an alternative method to detect and escape fruitless cycles is discussed. Basically it requires checking 2-cycles from time to time, while longer cycles are checked more rarely, because, according to [6], 2-cycles appear with the largest probability \((2r)^{-1}\). Bos, Kleinjung and Lenstra [2] have suggested the following heuristic: a cycle with at least one doubling is most likely not fruitless. Their main argument goes as follows. From a point on the cycle, \(R = aP + bQ\) say, the subsequent points are generated by adding one of the \(M_i = a_iP + b_iQ\) or by doubling, and negating if needed. If \(c \geq 1\) is the number of doublings during the whole cycle, then

\[
R = \pm 2^c R + \sum_{i=1}^{r} c_i M_i = \pm 2^c R + \sum_{i=1}^{r} c_i a_i P + \sum_{i=1}^{r} c_i b_i Q \text{ with } c_i \in \mathbb{Z}.
\]

Since \(1 \mp 2^c \neq 0\), then solving for \(Q\) the resulting equation\(^6\)

\[
\left((1 \mp 2^c)a - \sum_{i=1}^{r} c_i a_i\right)P + \left((1 \mp 2^c)b - \sum_{i=1}^{r} c_i b_i\right)Q = O
\]

might very likely yield a solution of the DLP (2), for the coefficient of \(Q\) is most likely not divisible by \(|G|\).

4 A new iterating function

Here we describe and briefly discuss a new function, continuing a study on the Pollard method initiated in [4] and [7]. Our function may be thought of as a sort of \(r + 1\)-mixed function, but it has two variables: the first one for an

\(^6O\) denotes the neutral element of \(G\). It is the point at infinity if \(G\) is an elliptic group.
element of an elliptic group and the other for the step number of the iteration.

More precisely, for a fixed integer \( r \geq 3 \), let the hash value \( \mathcal{H}(X) \) be the least positive residue mod \( r \) of the \( x \)-coordinate of \( X \in E(\mathbb{F}_q) \) and let \( M_1, \ldots, M_r \in E(\mathbb{F}_q) \) be preassigned\(^7\) such that \( \mathcal{H}(M_k) \equiv r \). Then, we set

\[
F_{\mathcal{H}}(X, i) := \begin{cases} 
X + M_{\mathcal{H}(X) + i} & \text{if } i \not\equiv 0 \pmod{r} \\
2X & \text{if } i \equiv 0 \pmod{r}
\end{cases},
\]

and from a starting pair \((X_0, 1)\) with a randomly chosen \( X_0 \in E(\mathbb{F}_q) \), we write the initial \( r + 1 \) steps of the iterating sequence \( X_{i+1} := F_{\mathcal{H}}(X_i, i + 1) \) as

\[
X_0 \xrightarrow{X_0 + M_{\mathcal{H}(X)+1}} X_1 \xrightarrow{X_1 + M_{\mathcal{H}(X)+2}} X_2 \rightarrow \cdots \rightarrow X_i \xrightarrow{X_i + M_{\mathcal{H}(X)+i+1}} X_{i+1} \rightarrow \cdots
\]

\[
\cdots \rightarrow X_{r-2} \xrightarrow{X_{r-2} + M_{\mathcal{H}(X)+r-1}} X_{r-1} \xrightarrow{2X_{r-1}} X_r \xrightarrow{X_r + M_{\mathcal{H}(X)+r-1}} X_{r+1} \cdots
\]

where \( x_i := \mathcal{H}(X_i) \) for short. In general, for any \( i \geq 0 \) and \( s \geq 1 \) one has

\[
X_{i+s} = X_i + \sum_{j=1}^{i+s-1} Y_j
\]

with \( Y_j := \begin{cases} 
M_{x_j+j+1} & \text{if } j + 1 \not\equiv 0 \pmod{r} \\
x_j & \text{if } j + 1 \equiv 0 \pmod{r}
\end{cases}. \tag{3}
\]

Next theorem shows that the period of \( X_j \) is necessarily a multiple of \( r \).

**Theorem 4.1.** The following two implications are true.

\[
X_{i+s} = X_i \text{ for some } s \equiv r \implies X_{i+s+k} = X_{i+k} \text{ for every } k \geq 0, \tag{4}
\]

\[
X_{i+s} = X_i \text{ for some } s \not\equiv r, 0 \implies X_{i+s+1} \neq X_{i+1}. \tag{5}
\]

**Proof.** From (3) we get \( X_{i+s} = X_i \iff \Delta_s(X_i) := \sum_{j=i}^{i+s-1} Y_j = O. \)

Further, it is easy to see that \( X_{i+s} = X_i \) for some \( s \equiv r, 0 \) implies that \( Y_{i+s} = Y_i \). Since it turns out that \( \Delta_s(X_{i+1}) = \Delta_s(X_i) + Y_{i+s} - Y_i \), then

\[
X_{i+s} = X_i \text{ for some } s \equiv r, 0 \implies X_{i+s+1} = X_{i+1}.
\]

Hence, (4) follows by induction on \( k \).

Let us assume that (5) is not true for some \( s \not\equiv r, 0 \) and some \( i \geq 0 \), i.e.

\[
\Delta_s(X_i) = \Delta_s(X_{i+1}) = O, \tag{6}
\]

that in turn implies \( Y_{i+s} = Y_i \). Since \( s \not\equiv r, 0 \), the possible cases are:

1) \( i + 1 \not\equiv r, 0 \), \( i + s + 1 \not\equiv r, 0 \);

2) \( i + 1 \not\equiv r, 0 \), \( i + s + 1 \equiv r, 0 \);

3) \( i + 1 \equiv r, 0 \).

Let us show that (6) leads to a contradiction in all the cases.

1) First note that the following two implications hold:

\[
i + 1 \not\equiv r, 0 \implies Y_i = M_{x_i+i+1},
\]

\[
i + s + 1 \not\equiv r, 0 \implies Y_{i+s} = M_{x_{i+s}+i+s+1} = M_{x_{i+i+s+1}},
\]

\(^7\)As usual, we also assume that the points \( M_i \) are randomly generated and that there are no trivial relations between them. Hereafter, the index \( k \) of \( M_k \) is always reduced mod \( r \).
where the last equality follows from $X_{i+s} = X_i$. Hence, from (6) we infer that $M_{x_i+i+1} = Y_i = Y_{i+s} = M_{x_i+i+s+1}$, in contradiction with $s \not\equiv r 0$.

2) As before one has $Y_i = M_{x_i+i+1}$, while $i + s + 1 \equiv r 0$ yields $Y_{i+s} = X_{i+s}$.

Since from (6) it follows $M_{x_i+i+1} = Y_i = Y_{i+s} = X_{i+s}$, then one gets $\mathcal{H}(X_i) = \mathcal{H}(M_{x_i+i+1}) \equiv r x_i + i + 1 \equiv r \mathcal{H}(X_i) + i + 1$, against $i + 1 \not\equiv r 0$.

3) Note that $i + 1 \equiv r 0$ imply $i + s + 1 \not\equiv r 0$ for $s \not\equiv r 0$. Thus, in this case (6) yields $X_{i} = Y_i = Y_{i+s} = M_{x_i+i+s+1} = M_{x_i+s}$, yet a contradiction with $s \not\equiv r 0$.

This concludes the proof of the Theorem.

Let us take $s = kr$ in (3) for some $k \geq 1$ and let $h_i \in \{0, \ldots, r-1\}$ be such that $h_i + i + 1 \equiv r 0$, i.e. $h_i + i + 1 = hr$ for some $h \geq 1$. Thus, we write

$$X_{i+kr} = X_i + \Delta_{h_i}(X_i) + (ps) + \Delta_{r-1}(X_{hr}) + (ps) + \Delta_{r-1}(X_{(h+1)r}) + (ps) + \cdots + \Delta_{r-1}(X_{(h+k-2)r}) + (ps) + \Delta_{r-h_i-1}(X_{(h+k-1)r}),$$

where $\Delta_{h_i}(X_i) = 0$ if $h_i = 0$ and in (ps) one has to add the previous summands. In case of a collision $X_{i+kr} = X_i$, without loss of generality we may assume that $i + 1 \equiv r 0$, because (4) implies that $X_{i+h_i+kr} = X_{i+h_i}$. Therefore, we take $h_i = 0$ and $i + 1 = hr$ in the above formula to see that

$$X_i = 2^k X_i + \sum_{j=1}^{k} 2^{k-j} \Delta_{r-1}(X_{i+1+(j-1)r}).$$

(7)

The amount of all the possible sums $\Delta_{r-1}(\cdot)$ in (7) is the number of the $(r-1)$-combinations with repetitions from the set of the $r$ points $M_i$, i.e.

$$\frac{(2k-2)!}{(r-1)!^2} \sim \frac{4^{r-1}}{\sqrt{\pi(r-1)}}.$$  

The expression (7) reminds the formula that we quote from [2] in §3. When using a mixed function, the heuristic argument in [2] suggests that "cycles occur only between two doublings. If the doubling frequency is sufficiently high, only short cycles would have to be dealt with". With our iterating function there is no need to check for $t$-cycles with $t < r$. Moreover, the formula (3) suggests that each of the $r$ subsequences $\{X_j\}_{j \mod r}$ can be regarded as a standard Pollard function, where $\Delta_r(X_j)$ is the rule that at each step prescribes a succession of $r$ addition (one of which is a doubling). Being numerous, such rules reduce the probability of the cycles in $\{X_j\}_{j \mod r}$ according to [6]. From this point of view, our new function behaves like a classical iterating function of one variable. In this way, $r$ valid points might be generated so that hopefully a distinguished couple of them might be useful to solve the ECDLP. We are going to prepare experimental tests in order to validate such speculations.

Acknowledgements. The authors wish to emphasize the excellent and constructive attention received from the referee.
References

[1] D.J. Bernstein, T. Lange, P. Schwabe, On the correct use of the negation
map in the Pollard rho method, *IACR Cryptology ePrint Archive*, 2011

the Pollard rho method, *Lecture Notes in Computer Science*, 6197 (2010),
66 - 82.

Solving a 112-bit prime elliptic curve discrete logarithm problem on game

based implementation of parallelized Pollard’s rho algorithm for ECDLP,
*Proceedings of the FINAL WORKSHOP OF PROJECTS FUNDED BY
"PON RICERCA 2000-2006, AVVISO 1575", ed. Consorzio COMETA*,


[6] I.M. Duursma, P.Gaudry and F. Morain, Speeding up the discrete log
computation on curves with automorphisms, *Lecture Notes in Computer
Science*, 1716 (1999), 103 - 121.


ematics of Computation*, 32 (1978), 918-924.

[9] E. Teske, Speeding up Pollard’s rho method for computing discrete loga-

(2001), 809 - 825.

survey), *In: Public-Key Cryptography and Computational Number The-


Received: May 12, 2014