Finding discrete logarithms with a set orbit distinguisher

Robert P. Gallant
Memorial University (SWGC)
rpgallant@swgc.mun.ca

June 28, 2010

Abstract

We consider finding discrete logarithms in a group $\mathbb{G}$ when the help of an algorithm D that distinguishes certain subsets of $\mathbb{G}$ from each other is available. For a group $\mathbb{G}$ of prime order p, if algorithm D is polynomial-time with complexity $c(\log(p))$, we can find discrete logarithms faster than square-root algorithms. We consider two variations on this idea and give algorithms solving the discrete logarithm problem in $\mathbb{G}$ with complexity $\mathcal{O}(p^{\frac{1}{3}}\log(p)^3 + p^{\frac{1}{3}}c(\log(p))$ and $\mathcal{O}(p^{\frac{1}{4}}\log(p)^3 + p^{\frac{1}{4}}c(\log(p))$ in the best cases. When multiple distinguishers are available logarithms can be found in polynomial time. We discuss natural classes of algorithms D that distinguish the required subsets, and prove that for *some* of these classes no algorithm for distinguishing can be efficient. The subsets distinguished are also relevant in the study of error correcting codes, and we give an application of our work to bounds for error-correcting codes.

AMS classification: 11T71, 94A60,11Y16

Keywords: Discrete logarithm problem, complexity, sparse polynomial, quadratic residue codes.

This research was supported by NSERC grant 327473-06.

1 Introduction

Let $\mathbb{G}$ be a group with generator G of prime order p. Since G generates $\mathbb{G}$, for any element $W \in \mathbb{G}$ there is a unique integer $w \in [0, p-1]$ such that $W = G^w$. The integer w is said to be the discrete logarithm of W to the base G. An algorithm computing the discrete logarithm of W to the base G, for a randomly chosen element $W \in \mathbb{G}$, is said to solve the discrete logarithm problem (DLP) in $\mathbb{G}$. The DLP is fundamental in cryptography, see [12, 16] for example. In cryptographic protocols based on the DLP one uses a group where it is assumed that solving the DLP is computationally difficult. For such protocols the group is usually a prime-order multiplicative subgroup of a finite field or a prime order subgroup of an elliptic curve group, and a standard assumption in these groups is that algorithms solving the discrete logarithm problem require at least $\Theta(\sqrt{p})$ bit operations.[1]

In Section 2 we outline algorithms for computing discrete logarithms. The theme in each case is that given a polynomial-time algorithm D that can distinguish the orbits of a prime-order group $\mathbb{G}$ (under exponentiation) from each other, we can find discrete logarithms in less than $\Theta(\sqrt{p})$ operations.

It is easy enough to construct algorithms that distinguish the required orbits. What is not clear is how small the complexity of such algorithms may be. A lower bound on the complexity of distinguishing these orbits could be obtained from assuming a lower bound on the complexity of solving the the discrete logarithm problem, and using the algorithms of this paper as a reduction. Such an approach is taken in [6], for example. Similarly the algorithms in this paper imply that an assumed lower bound of $\Theta(\sqrt{p})$ for the complexity of the discrete logarithm problem implies a complexity of at least $\Theta(\sqrt{p})$ for any such algorithm D. We discuss a polynomial reduction also. In Section 3, our main focus is to determine lower bounds on the complexity of such an algorithm D, under assumptions about the type of algorithm, instead of assumptions about the difficulty of the DLP. There we discuss natural approaches to constructing such algorithms, such as evaluating indicator polynomials. We give lower bounds on the sparsity of the relevant polynomials.

The orbits we consider, and polynomials zero on these orbits, are connected with error correcting codes. In [4] a generalization of quadratic residue codes is considered, and a bound on the minimal distance of such codes is given. There the author states "Ideally, we would like to see this qth-root bound hold for all codewords ...". In Section 3.3 we show how our bound from Section 3.2 can be interpreted as such a qth-root bound.

Many works study the connections between problems and assumptions used in cryptography, such as relating the complexity of the Diffie-Hellman Problem to that of the DLP (See [9, 11] for example.) Works such as [7, 10] consider polynomials solving cryptographic problems and bound their degree or sparsity, as do the results of this paper.

The One-More Diffie-Hellman Problem [1] (see [11] also) and the Strong Diffie-Hellman Problem [2], are variants of the standard Diffie-Hellman Problem where certain additional information is also available, and in the same sense we are considering the discrete logarithm problem when additional information (the distinguisher algorithm) is available.

The algorithms in [6, 5] improve square-root algorithms for finding discrete logarithms when additional information such as a 'static Diffie-Hellman' oracle is available. In the mindset of this paper, those algorithms are given a discrete logarithm challenge W and use the additional information to first compute a power of W, where the power in the same orbit (under exponentiation by a particular element $\alpha \in \mathbb{F}_p^*$) as the group generator. Thus these papers can be

[1]Even for subgroups of finite fields, where subexponential-time index calculus techniques are available, the sizes of the field and subgroup are often chosen so that this assumption is reasonable.

viewed as special case of what we consider here, and so the distinguishers we consider here are fundamentally related to the discrete logarithm problem. An important distinction between those works and ours is that such distinguishers help find logarithms of arbitrary elements in the group, whereas with a static Diffie-Hellman oracle, the oracle 'knows' just one specific logarithm and is not useful for finding other logarithms.

2 Finding Discrete Logarithms using an Orbit Distinguisher

For concreteness, we assume the group $\mathbb{G}$ in which we are finding discrete logarithms is a multiplicative subgroup of a finite field, though the algorithms of this section apply to any group. In the case of elliptic curve groups pairings can be used to map the problem of distinguishing orbits to the case we consider.

We begin by stating the standard assumptions that will be used in the paper.

Assumptions 1. *Assume $\mathbb{G}$ is a multiplicative subgroup of the finite field $\mathbb{F}_q$, with generator G of prime order p. Further assume $p = AB + 1$, for integers $A, B \geqslant 2$, with $\gcd(A, B) = 1$. Assume $w \in \mathbb{F}_p$ is primitive, and that $\alpha = w^B$ and $\beta = w^A$.*

Element $\alpha \in \mathbb{F}_p^*$ has multiplicative order $\operatorname{ord}_p(\alpha) = A$, and similarly $\operatorname{ord}_p(\beta) = B$. Because $\gcd(A, B) = 1$, the elements α and β generate $\mathbb{F}_p^*$, so any element of $\mathbb{F}_p^*$ can be written as $\alpha^i\beta^j$ for appropriate integers i, j. Since $\mathbb{G}$ has order p, exponentiation of elements in $\mathbb{G}$ by elements of $\mathbb{F}_p$ is well-defined. Thus, $\mathbb{G}$ consists of the identity element '1' along with the elements in the set $\{G^{\alpha^i\beta^j} | 0 \leqslant i < A, 0 \leqslant j < B\}$.

Consider the action on $\mathbb{G}$ of the function taking $x \to x^\beta$. The image of element $G^{\alpha^i\beta^j}$ is $G^{\alpha^i\beta^{j+1}}$, and as $\beta^B = 1$, we see for each i, $0 \leqslant i < A$, the set $O_i = \{G^{\alpha^i\beta^j} | 0 \leqslant j < B\}$ is an orbit under the action of this function. The orbits partition $\mathbb{G}$; there are A orbits of size B, and one orbit of size 1 (the orbit containing the identity element '1' in $\mathbb{G}$). Raising element $G^{\alpha^i\beta^j}$ in orbit O_i to a power $\alpha^x\beta^y$ (an element of $\mathbb{F}_p^*$) will result in an element in orbit $O_{i+x \bmod A}$, which will be a different orbit if x is nonzero modulo A. Later we use the following consequence of this.

Lemma 2. *If $\lambda \in \mathbb{F}_p^*$ has $A \mid \operatorname{ord}_p(\lambda)$, and R in $\mathbb{G}$ has order p, then no two of the elements $R, R^\lambda, R^{\lambda^2}, \ldots, R^{\lambda^{A-1}}$ are in the same orbit O_i, and each orbit of size B contains exactly one of these elements.*

Proof. Suppose $\lambda = \alpha^i\beta^j$ and $R = G^{\alpha^a\beta^b}$. If R^{λ^u} and R^{λ^v} are in the same orbit, then $\alpha^a\alpha^{iu} = \alpha^a\alpha^{iv}$, so that $i(u-v) = 0 \bmod A$. As $\operatorname{ord}_p(\lambda)$ is a multiple of A, we have $\gcd(i, A) = 1$, and thus $u = v \bmod A$. $\square$

The orbits O_i do not depend on the particular element of order B we use for exponentiation, since any element of order B is a power of any other element of order B in $\mathbb{F}_p^*$. For the rest of this paper, by 'B-orbit of $\mathbb{G}$' we mean an orbit of $\mathbb{G}$ of size B under the action of exponentiation by an element β in $\mathbb{F}_p^*$ of order B as described above. We will sometimes say 'orbit' when the value B and group $\mathbb{G}$ is clear.

We now consider two different ways that an algorithm might distinguish the orbits of $\mathbb{G}$ from each other, and in each case exploit this to find discrete logarithms in $\mathbb{G}$.

2.1 Type-1 Distinguisher: indicator for a single orbit

Suppose for a fixed orbit S of $\mathbb{G}$ there exists an algorithm D such that on input element $X \in \mathbb{G}$, the algorithm D

1. outputs 1 if $X \in S$, and
2. outputs 0 if $X \notin S$.

We will call such an algorithm a type-1 distinguisher for $\mathbb{G}$. It is simply an indicator function for the orbit S.

Observe that if one is given such an algorithm D that determines whether an input X in the group is in *one particular* orbit of S, then, fixing an exponent e at random, and forming the algorithm that on input X returns $D(X^e)$, one obtains an algorithm that distinguishes a random orbit of $\mathbb{G}$.

Given a type-1 distinguisher D for $\mathbb{G}$, we will want to know 'which' orbit it distinguishes, in that we we want to know, for a given generator G of $\mathbb{G}$, the unique integer a in $[0..A-1]$ such that $D(G^{\alpha^a}) = 1$. It is possible to find which orbit is distinguished by checking each $D(G^{\alpha^i})$, for $i = 0 \ldots A-1$. In the worst case this will take A calls to the distinguisher.

We will assume function c is an upper bound on the time-complexity of D, meaning that when D is given an input of length l then D requires at most $c(l)$ operations to complete. In particular, when $\mathbb{G}$ has order p we assume that evaluating D at any element $X \in \mathbb{G}$ requires at most $c(\log(p))$ bit operations. We will say D is a polynomial-time algorithm if there exists a polynomial h such that $c(l) \leqslant h(l)$ for all allowable input lengths l.[2]

The following algorithm takes as input $W \in \mathbb{G}$, and outputs the integer $w \in [0, p-1]$ such that $G^w = W$. The loops in steps 2 and 3 serve to place a power G' of G and a power W' of W in the orbit S distinguished by D, and then in step 4 a Baby-Step-Giant-Step search is applied inside S to find the logarithm of W' to the base G'.

Algorithm 1, with inputs $(W, G, p, \alpha, A, \beta, B)$ and which may evaluate D:

1. If $W = 1$ then the algorithm outputs $w = 0$ and exits.

[2]To use the notion of polynomial-time, we assume algorithm D distinguishes an orbit in a group of size p for infinitely many primes p. If one has such a distinguisher for only a single group, the algorithm here may still be of interest provided the computational cost of the distinguisher is less than the cost of $p^{1/6}$ exponentiations in the group.

2. For each $u \in [0, 1, \ldots, A-1]$:
   - If $D(G^{\alpha^u}) = 1$ then goto 3
3. For each $v \in [0, 1, \ldots, A-1]$:
   - If $D(W^{\alpha^v}) = 1$ then goto 4
4. Set $J = \lceil \sqrt{B} \rceil$. Set $G' = G^{\alpha^u}$, $W' = W^{\alpha^v}$. Find an element from the sets
$$\{(G'^{\beta^{iJ}}, i) | i = 0 \ldots J-1\}$$
and
$$\{(W'^{\beta^{-i}}, i) | i = 0 \ldots J-1\}$$
with first coordinates equal (as group elements). The second coordinates provide integers m, n such that $G'^{\beta^{mJ}} = W'^{\beta^{-n}}$.
5. Thus $G^{\alpha^u \beta^{mJ}} = W^{\alpha^v \beta^{-n}}$, which gives the logarithm of W to the base G, namely $\frac{\alpha^u \beta^{mJ}}{\alpha^v \beta^{-n}} \pmod{p}$, which is output by the algorithm.

The computations required by this algorithm are at most: $2A$ exponentiations of elements of $\mathbb{G}$ to the power α, $2J$ exponentiations of elements of $\mathbb{G}$ to the power β, $2A$ evaluations of D, $2A{+}1$ comparisons, the cost of finding elements from two sets each of size at most J with a common first coordinate (which can be done using merge sort with at most $\mathcal{O}(J \log(J))$ comparisons of group elements), and a constant number (independent of p) of multiplications in $\mathbb{F}_p^*$. Assuming each exponentiation can be done in at most $\mathcal{O}(\log(p)^3)$ bit operations, and recalling that $J = \lceil \sqrt{B} \rceil$, we find this algorithm thus has complexity $\mathcal{O}(A \log(p)^3 + \sqrt{B} \log(p)^3 + Ac(\log(p)))$.

If both A and $\sqrt{B}$ are both approximately $p^{1/3}$, which can happen if $p-1$ factors appropriately, then the algorithm complexity is $\mathcal{O}(p^{1/3} \log(p)^3 + p^{1/3} c(\log(p)))$, which, if D is a polynomial-time algorithm, is asymptotically less than $p^{1/2}$.

If c is an exponential function in $\log(p)$, it is still possible for this algorithm to be asymptotically smaller than $p^{1/2}$, for example, $A \approx p^{1/3}$, $B \approx p^{2/3}$, and $c(\log(p)) \approx p^{1/7}$. Thus there may be distinguisher algorithms D with non-polynomial complexity for which this algorithm finds logarithms faster than square-root algorithms.

Clearly the use of precomputed powers of G and W can improve the logarithmic factors in the algorithm. See Appendix A for a technique for dealing with the large storage requirement.

Is it possible to have a polynomial-time algorithm D with the required properties? In Section 3 we discuss these issues further, but the "devil's advocate" might argue the affirmative as follows. For a coset of a subgroup of order p in $\mathbb{F}_q^*$, there is a polynomial of the form $x^p - C$ that is zero on the coset and nonzero at other elements of $\mathbb{F}_q^*$. Thus one can efficiently distinguish one coset from the others. Arguably, the special property of the coset that allows for such an easily-evaluated polynomial to test for inclusion in the coset is the closure of the subgroup. But the orbits of $x \to x^\beta$ on $\mathbb{G}$ have a similar closure property (closure under exponentiation by the power β) so perhaps this feature can be exploited to efficiently distinguish these orbits.

With the exception of the distinguisher D, the algorithm above is generic, in the sense of [15]. One can implement a distinguisher D that is also generic, and (unsurprisingly) the natural way gives a distinguisher with complexity $\mathcal{O}(\sqrt{B})$, and so we do not contradict the lower bound in [15].[3] In Section 3 we discuss the possibility of other, efficient, distinguishers; for example, perhaps a short straight-line program for the indicator polynomial of orbit S exists. However such a distinguisher would certainly use field addition as well as field multiplication, and so would require operations not available through a generic-group oracle, and so again we do not contradict [15].

2.2 Type-2 Distinguisher: identifies the the containing orbit

In this variation, we consider a more powerful, but still somewhat natural, way that the orbits might be distinguished. Here we consider distinguishers that can tell us 'which' orbit of size B contains a given element $X \in \mathbb{G}$.

Suppose there exists an algorithm D, taking as input an element of $\mathbb{G}$, and giving as output a bitstring such that

1. the outputs $D(u)$ and $D(v)$ are the same if u and v are in the same orbit, and
2. the outputs $D(u)$ and $D(v)$ are different if u and v are in different orbits.

We will call such an algorithm a type-2 distinguisher for $\mathbb{G}$.

As before we suppose evaluating algorithm D costs at most $c(\log(p))$. Whereas in Section 2.1 the algorithm D could distinguish one of the orbits from the others, here we assume D can distinguish any two orbits from each other, and in fact 'label' (with the bitstring output) the orbit in which an element lies, which is requiring considerably more of algorithm D.

The following algorithm takes as input $W \in \mathbb{G}$, and outputs the integer $w \in [0, p-1]$ such that $G^w = W$. It is similar to the previous algorithm, but leverages the fact that D can now label the orbit in which an element lies to perform an initial square-root search to find an orbit containing a power G' of G and a power W' of W (instead of the linear search used in the previous algorithm.)

Algorithm 2, with inputs $(W, G, p, \alpha, A, \beta, B)$ and which may evaluate D:

1. If $W = 1$ then the algorithm outputs $w = 0$ and exits.

[3]We may interpret the lower bound in [15] as a proof that in the generic group model no distinguisher can be polynomial-time.

2. Set $K = \lceil \sqrt{A} \rceil$. Find an element from the sets
$$\{(D(G^{\alpha^{iK}}), i) | i = 0 \dots K-1\}$$
and
$$\{(D(W^{\alpha^{-i}}), i) | i = 0 \dots K-1\}$$
with first coordinates equal. The second coordinates provide integers m, n such that $D(G^{\alpha^{mK}}) = D(W^{\alpha^{-n}})$, and therefore $G^{\alpha^{mK}}$ and $W^{\alpha^{-n}}$ are in the same orbit.

3. Set $J = \lceil \sqrt{B} \rceil$. Set $G' = G^{\alpha^{mK}}$, $W' = W^{\alpha^{-n}}$. Find an element from the sets
$$\{(G'^{\beta^{iJ}}, i) | i = 0 \dots J-1\}$$
and
$$\{(W'^{\beta^{-i}}, i) | i = 0 \dots J-1\}$$
with first coordinates equal (as group elements). The second coordinates provide integers u, v such that $G'^{\beta^{uJ}} = W'^{\beta^{-v}}$.

4. Thus $G^{\alpha^{mK}\beta^{uJ}} = W^{\alpha^{-n}\beta^{-v}}$, which gives the logarithm of W to the base G, namely $\frac{\alpha^{mK}\beta^{uJ}}{\alpha^{-n}\beta^{-v}} \pmod{p}$, which is output by the algorithm.

The computations required by Algorithm 2 are at most: $2K$ exponentiations of group elements to the power α, $2J$ exponentiations of group elements to the power β, $2K$ evaluations of D, the cost of finding the set elements with common first coordinates which is $\mathcal{O}(K \log(K) + J \log(J))$, and a constant number (independent of p) of multiplications in $\mathbb{F}_p^*$. As before, assuming the exponentiations cost $\mathcal{O}(\log(p)^3)$ we find the complexity of Algorithm 2 is thus $\mathcal{O}(\sqrt{A}\log(p)^3 + \sqrt{B}\log(p)^3 + \sqrt{A}c(\log(p)))$.

If A and B are both about $p^{1/2}$, the complexity of Algorithm 2 is $\mathcal{O}(p^{1/4}\log(p)^3 + p^{1/4}c(\log(p))$. If D is a polynomial-time algorithm, then Algorithm 2 has complexity smaller than $p^{1/2}$, though as before it is possible for this to occur with c an exponential function also. (For example, $A \approx p^{1/2}, B \approx p^{1/2}$, and $c(\log(p)) = p^{1/5}$.)

2.3 Multiple distinguishers give a polynomial reduction to DLP

In the previous sections, we considered having access to only *one* distinguisher for an orbit under a single given exponent β. In this section, we consider the case that $p-1$ is smooth, and for several distinct values of B dividing $p-1$, we have a type-1 distinguisher for an orbit of length B in $\mathbb{G}$. In this case we can find discrete logarithms in $\mathbb{G}$ in a polynomial number of steps.

Lemma 3. *Suppose p is prime with $p-1 = AB$, and that w is primitive in $\mathbb{F}_p$, and $\alpha = w^B$, $\beta = w^A$, and that G generates a group $\mathbb{G}$ of order p in $\mathbb{F}_q$. Suppose $X = G^{w^x}$, and that X^{α^d} and G^{α^c} are in the same length-B orbit of $\mathbb{G}$ under exponentiation to power β. Then* $x = B(c-d) \bmod A$.

Proof. Since $(G^{w^x})^{\alpha^d}$ and G^{α^c} are in the same length-B orbit, there is an integer i such that $G^{w^x \alpha^d} = (G^{\alpha^c})^{\beta^i}$ and so $w^x \alpha^d = \alpha^c \beta^i$ (in $\mathbb{F}_p$), and so $w^x = \alpha^{c-d}\beta^i$, and thus $w^x = w^{B(c-d)+Ai}$ and so, as w has order $p-1 = AB$, we have $x = B(c-d) + Ai \bmod AB$ and thus $x = B(c-d) \bmod A$. $\square$

Theorem 4. *Suppose $G \in \mathbb{F}_q$ generates a group $\mathbb{G}$ of prime order p, and that w is primitive in $\mathbb{F}_p$, and that $p-1 = A_1A_2A_3...A_k$, with the A_i pairwise relatively prime, and for each $B_i = (p-1)/A_i$ we have a B_i-orbit (type-1) distinguisher D_{B_i}. Then given an arbitrary element X in $\mathbb{G}$, $X \neq 1$, there is an algorithm to find an integer x satisfying $X = G^{w^x}$ requiring at most $2A_i$ calls to each distinguisher D_{B_i}, and a number of bit operations and group multiplications that are at most a polynomial in* $\log(p)$.

Proof. Consider the following algorithm which takes as input $(G, p, w, A_1, A_2, \ldots, A_k, X)$ and which may evaluate the distinguishers $D_{B_1}, D_{B_2}, \ldots, D_{B_k}$:

1. for $i = 1, 2 \ldots, k$ do
   (a) Set $\alpha_i = w^{B_i}, \beta_i = w^{A_i}$.
   (b) Determine c_i such that $D_{B_i}(G^{\alpha^{c_i}}) = 1$
   (c) Determine d_i such that $D_{B_i}(X^{\alpha^{d_i}}) = 1$. Observe that since $G^{\alpha^{c_i}}$ and $X^{\alpha^{d_i}}$ are in the same orbit, the previous lemma gives $x = B_i(c_i - d_i) \bmod A_i$.
2. Use the Chinese Remainder Theorem to explicitly find the unique positive integer $x \in [0, p-1]$ such that $x = B_i(c_i - d_i) \bmod A_i$ for $i \in 1, \ldots, k$.
3. Output this x.

The complexity claims are easily verified.

$\square$

Clearly an algorithm that can find discrete logarithms in $\mathbb{G}$ can be used to efficiently construct a distinguisher D_{B_i}. Thus this theorem can be viewed as a polynomial-time reduction between the problem of distinguishing the B_i-orbits and finding discrete logarithms in $\mathbb{G}$.

2.4 A connection to a conjecture of Boneh and Lipton

In the previous section we considered having access to multiple distinguishers D_{B_i}. In this section we discuss whether having access to a single distinguisher can give a polynomial time algorithm for finding discrete logarithms.

A type-1 distinguisher D identifies when an element $X \in \mathbb{F}_q$ is in a particular B-orbit, say the orbit consisting of elements $G^{\alpha^a \beta^j}$, for $j = 0 \ldots B-1$. As noted earlier to determine the value of a we can evaluate the distinguisher at A elements of the form G^{α^i}. When X is in the distinguished orbit, the element $x \in \mathbb{F}_p$ satisfying $X = G^x$ can be written as $x = \alpha^a \beta^j$ for some j, and so as β

has order B this implies the value x is a root of $t^B - \alpha^{aB} = 0$; namely when $t = x$ this equation is satisfied.

It may be helpful to interpret the previous paragraph as follows: $D(X) = 1$ when the discrete logarithm (to the base G) of X is in a particular coset of the order B subgroup of $\mathbb{F}_p^*$, namely the same coset of the subgroup $\langle\beta\rangle$ as the element α^a.

We continue by focusing on a special case. Suppose $B = (p-1)/2$ and D is a type-1 distinguisher for the orbit consisting of elements $G^{\alpha^0\beta^j}$, $j = 0\ldots, B-1$, and that for an element $X \in \langle G\rangle$, we have $D(X) = 1$. This tells us that the unique $x \in \mathbb{F}_p$ satisfying $X = G^x$ has the property that it can be written as $x = \alpha^0\beta^j$ for some j; or in other words that $x = \beta^j$ for some j, which implies that $x^B - 1 = 0$ since β has order B. In this case this means (since $B = (p-1)/2$) that the discrete logarithm x is a quadratic residue in $\mathbb{F}_p$. So this distinguisher determines whether the discrete logarithm (to the base G) of its input X is a quadratic residue in $\mathbb{F}_p$.

Similarly, suppose also that the product $XG \neq 1$ and $D(X \cdot G) = 0$; then we know $XG = G^{x+1}$ must lie in the orbit $G^{\alpha^1\beta^j}$ (since in this case there are only two orbits of length B) and so $(x+1)^B - \alpha^B = 0$. In this case α has order $A = 2$, and $\gcd(A, B) = 1$ so B is odd and so the equation is simply $(x+1)^B + 1 = 0$. In other words $x + 1$ is a quadratic non-residue in $\mathbb{F}_p$.

We could continue in this way (determining the orbit containing $X^2G, X^3G, \ldots$) to find several polynomials (of degree $(p-1)/2$, but that have very short straight-line programs) which all have x as a root. If it were possible to easily find this common root (by taking gcd's, for example) we would have the desired logarithm x.

But no method for efficiently finding the common root is known in this scenario. A similar situation is discussed in [3], where the authors define the signature of $x \in \mathbb{F}_p$ as the vector

$$sig(x) = \left(\left(\frac{x}{p}\right), \left(\frac{x+1}{p}\right), \ldots, \left(\frac{x+k}{p}\right)\right)$$

where $k = \lceil 2\log^2 p\rceil$, and conjecture that for sufficiently large p, any two distinct x, y have $sig(x) \neq sig(y)$. They also state that there is no known polynomial time algorithm for finding x given $sig(x)$.

In summary, in Section 2 we considered how algorithms capable of distinguishing orbits of group $\mathbb{G}$ under exponentiation can help solve the discrete logarithm problem in $\mathbb{G}$. We now turn our attention to direct consideration of how difficult it might be to find such distinguishers.

3 Distinguishing Orbits Efficiently

The algorithms in the previous section required an algorithm D that distinguished the orbits of $\mathbb{G}$ under the action of exponentiation by a fixed integer. In this section we discuss *some* natural ways one might implement such an algorithm, and in a few cases give lower bounds on the resulting complexity.

3.1 An indicator polynomial for an orbit

Given any B-orbit S in $\mathbb{G}$, the polynomial $I_S(x) = \prod_{r \in S}(x - r)$ is zero on S and nonzero on $\mathbb{G} \setminus S$. For any polynomial $h(x) \in \mathbb{F}_q[x]$ with no zeros in $\mathbb{G} \setminus S$, the polynomial $f(x) = h(x)I_S(x)$ is also zero on S and nonzero on $\mathbb{G} \setminus S$. An algorithm determining whether such a polynomial is nonzero at a point (such as 'evaluate the polynomial using Horner's rule and test if the result equals zero') is thus a type-1 distinguisher, as in Section 2.1. To be useful for finding discrete logarithms, the orbit size would be rather large, for example approximately $p^{2/3}$. In this case the degree of f will be large. Can any such polynomial be evaluated efficiently, that is, in time at most some polynomial in $\log(p)$?

This can occur if f has relatively few nonzero coefficients, but it is unclear whether one can find an orbit S and a polynomial h such that $h(x)I_S(x)$ is relatively sparse. In Section 3.2 we give a lower bound on the number of nonzero coefficients in a polynomial that is zero on an orbit, and this lower bound partially settles this question.

Another possibility is that there are polynomials zero on S and nonzero on $\mathbb{G} \setminus S$ having many nonzero coefficients but that can still be evaluated efficiently. For example, perhaps a polynomial of this form can have a short straight-line program. This possibility seems open.

Given the A different orbits of $\mathbb{G}$ of size B, we can construct (using interpolation) many polynomials $D(x)$ such that $D(u) = D(v)$ if u and v are in the same orbit, and $D(u) \neq D(v)$ if u and v are in different orbits. Algorithms evaluating such polynomials are type-2 distinguishers. In Section 3.4 we show such a polynomial must have at least B nonzero coefficients. It is unknown whether a polynomial with this property can be represented by an efficient straight-line program.

The previous examples are straightforward ways one might try to construct a type-1 or type-2 distinguisher, but there are many other approaches. As a final example: perhaps one can find an efficient function that produces a square 2/3 of the time on some 'random' half of the orbits, and only 1/3 of the time on the other half. Such a function can be used to create an efficient type-2 distinguisher.

We are not suggesting that efficient versions of such distinguishers exist. We simply note that such distinguishers do not seem to have been considered before in this context, and that if it is possible for them be computationally efficient then there are implications for the discrete logarithm problem. We now show that *some* of the possible ways to implement distinguishers can never be efficient.

3.2 Sparsity of a polynomial zero on an orbit

In this section we give a lower bound on the number of nonzero coefficients in a polynomial that is zero on an orbit.

We use the following fact which we state without proof.

Lemma 5. *Let F be a field, and let p a positive integer. Let $f \in F[x]$ have the form $f(x) = \sum_{i=0}^{n} f_i x^i$. Then $f(x)$ is divisible by $(x^p - 1)$ if any only if for each integer $r \in [0, p-1]$ we have*

$$\sum_{\{j|j \equiv r \bmod p\}} f_j = 0.$$

Informally this says that the polynomial $f(x)$ is a multiple of $(x^p - 1)$ if and only if the polynomial resulting from reducing every exponent of x modulo p is the zero polynomial.

Theorem 6. *Assume $\mathbb{G}$ is a multiplicative subgroup of prime order p of the finite field $\mathbb{F}_q$, and that $p - 1 = AB$ for integers $A, B \geqslant 2$ with $\gcd(A, B) = 1$, and that integers α, β satisfy $1 < \alpha, \beta < p$ and $\operatorname{ord}_p(\alpha) = A$ and $\operatorname{ord}_p(\beta) = B$. Assume that $f \in \mathbb{F}_q[x]$ is zero on an orbit S of size B of the function $x \to x^\beta$ on group $\mathbb{G}$, but not zero everywhere on $\mathbb{G}$. Then the number of nonzero terms in $f(x)$ is at least $\sqrt[A]{\frac{\phi(p-1)}{A-1}}$ (with ϕ the totient function.)*

Proof. To begin we assume the polynomial f has degree less than p. Since f is not zero everywhere on $\mathbb{G}$, we can write $f(x) = \sum_{i=1}^{s} a_i x^{e_i}$, with each $a_i \neq 0$, and $s \geqslant 1$. By Lemma 2, for any $\lambda \in \mathbb{F}_p^*$ of order $p - 1$, and any element ζ in $\mathbb{G}$ of order p, one of $\zeta, \zeta^\lambda, \zeta^{\lambda^2}, \zeta^{\lambda^3}, \ldots, \zeta^{\lambda^{A-1}}$ is in S, and so the polynomial $w(x) = f(x)f(x^\lambda)f(x^{\lambda^2}) \ldots f(x^{\lambda^{A-1}})$ is zero at ζ. Since $w(x)$ is zero at each element of order p in $\mathbb{G}$, it must be a multiple of the polynomial $(x^p - 1)/(x - 1)$, and so the remainder of polynomial $w(x)$ upon division by $x^p - 1$ is either 0 or $c(1 + x + x^2 + \ldots x^{p-1})$, for some $c \in \mathbb{F}_q^*$.

Suppose for some $\lambda \in \mathbb{F}_p^*$ of order $p - 1$ the remainder is $c(1 + x + x^2 + \ldots + x^{p-1})$, for some $c \in \mathbb{F}_q^*$. The number of nonzero coefficients in the remainder of $w(x)$ upon division by $x^p - 1$ is at most the number of nonzero coefficients in $w(x)$ itself. Since $w(x)$ has at most s^A nonzero coefficients, and the remainder of $w(x)$ upon division has p nonzero coefficients, we have $s^A \geqslant p$ and so $s \geqslant \sqrt[A]{p} \geqslant \sqrt[A]{\phi(p-1)/(A-1)}$.

Otherwise suppose for every primitive $\lambda \in \mathbb{F}_p^*$ the remainder of $w(x)$ after division by $x^p - 1$ is 0. A typical monomial in the expansion of $w(x)$ is

$$a_{u_0} a_{u_1} \ldots a_{u_{A-1}} x^{e_{u_0} + e_{u_1}\lambda + \ldots e_{u_{A-1}}\lambda^{A-1}}$$

and is identified with the tuple $(u_0, u_1, \ldots, u_{A-1}) \in \{1, 2, \ldots, s\}^A$.

By Lemma 5, for any primitive $\lambda \in \mathbb{F}_p^*$ and $r \in \{0, 1, 2, \ldots p - 1\}$, the sum $S_{\lambda,r} = \sum a_{u_0} a_{u_1} \ldots a_{u_{A-1}}$ is zero, where the sum is over $(u_0, u_1, \ldots, u_{A-1}) \in \{1, 2, \ldots, s\}^A$ such that the equation $e_{u_0} + e_{u_1}\lambda + \cdots + e_{u_{A-1}}\lambda^{A-1} = r$ holds in $\mathbb{F}_p$. In particular, since each a_i is nonzero each sum either contains no terms at all or at least two terms.

We focus on those sums involving a particular term, say $a_1 a_1 \ldots a_1 = a_1^A$. We count the number N of pairs (λ, t) where λ is primitive in $\mathbb{F}_p^*$ and t is a tuple in the set $\{1, 2, \ldots, s\}^A \setminus \{\{1\}^A\}$, say $t = (u_0, u_1, \ldots, u_{A-1})$, that identifies a term $a_{u_0} a_{u_1} \ldots a_{u_{A-1}}$ occurring in some sum $S_{\lambda,r}$ with the term a_1^A.

For each primitive $\lambda \in \mathbb{F}_p^*$, the term a_1^S occurs in some sum $S_{\lambda,r}$, and that sum contains at least one other term, so the number of pairs N is at least $\phi(p-1)$.

How often can a given a given tuple $(u_0, u_1, \dots, u_{A-1})$ from $\{1, 2, \dots, s\}^A \setminus \{\{1\}^A\}$ occur in such a pair? If this tuple occurs in a sum $S_{\lambda,r}$ with a_1^A, then we have $e_{u_0} + e_{u_1}\lambda + \dots e_{u_{A-1}}\lambda^{A-1} = e_1 + e_1\lambda + \dots + e_1\lambda^{A-1}$, and there are at most $A-1$ such values λ. Hence the number of pairs $N \leqslant (s^A - 1)(A-1)$.

Thus we have $\phi(p-1) \leqslant (s^A - 1)(A-1)$ which gives the bound in this case.

Finally, given a polynomial f of arbitrary degree that is zero on an orbit S of size B, the remainder R of this polynomial after division by $x^p - 1$ has $f(r) = R(r)$ for each $r \in \mathbb{G}$, and R has no more nonzero terms than does f. So the remainder polynomial R is also zero on orbit S, and is not zero everywhere on $\mathbb{G}$, and has degree less than p, and so the above argument gives a lower bound on the number of nonzero terms in R, and this is also a lower bound on the number of nonzero terms in f. $\square$

If, as in the theorem statement, a polynomial f with s nonzero coefficients is zero on an orbit under exponentiation by an element β of order B in $\mathbb{F}_p^*$, and if hB also divides $p-1$, then one can use f to construct a polynomial having at most s^h nonzero coefficients that is zero on an orbit under exponentiation by an element of order hB. So the number of nonzero coefficients in this new polynomial is bounded by the theorem. One may use this idea to obtain a minor improvement in the bound of the theorem, where A in the denominator is replaced by the smallest prime-power dividing A.

One could also obtain a minor improvement to the bound by considering those λ with order a multiple of A instead of just primitive λ, as we have in the proof.

We must have A less than than $\log(p)$ for this bound to rule out the possibility of a sparse polynomial that is zero on an orbit giving rise to an efficient distinguisher, which means the bound is of limited use in this application. In particular, the bound does not rule out the cryptographically interesting possibility of a polynomial having few nonzero coefficients, and that is zero on an orbit of size approximately $p^{2/3}$. However in the next section we show how this this bound gives a connection between the discrete logarithm problem and error correcting codes.

3.3 Connection to quadratic residue codes

Theorem 6 provides a lower bound on the number of nonzero coefficients in a polynomial that is zero on an orbit under exponentiation. Such polynomials occur in the study of error correcting codes.

For our purposes, a quadratic residue code may be defined as follows. Assume p, l are primes with p odd and l a quadratic residue modulo p, and that $\mathbb{F}_q$ is a finite extension of $\mathbb{F}_l$ containing a primitive pth root of unity r, and that Q is the set of quadratic residues modulo p. Then the polynomial $g(x) = \prod_{e \in Q}(x - r^e)$ is in $\mathbb{F}_l[x]$. Polynomials in $\mathbb{F}_l[x]$ that are a multiple of

$g(x)$ have the common set of zeros $\{r^e | e \in Q\}$. Such polynomials that also have degree less than p are the quadratic residue code of length p over $\mathbb{F}_l$. The minimum number of nonzero coefficients among such polynomials is the distance of the code, and so any bound on the minimum number of nonzero coefficients in such polynomials is a bound on the minimum distance of the code.

The common root set $\{r^e | e \in Q\}$ of these polynomials is an orbit in under exponentiation by an element $\beta \in \mathbb{F}_p^*$ of order $(p-1)/2$. Thus Theorem 6 can be applied. In this case, one obtains the well-known square-root bound for such codes [8]. Indeed, the first part of the proof of Theorem 6 is a typical proof of the square-root bound.

In [4] a generalization of quadratic residue codes called lth-power residue codes is considered, and a lower bound on their minimum distance is proved.

That generalization is as follows, though we point out that our description differs somewhat, and in particular, the role of q in [4] is played by l in the description below. Assume primes p, l with $l | p-1$, and assume l is an lth power residue mod p. Suppose w generates $\mathbb{F}_p^*$. Partition $\mathbb{F}_p^*$ into sets A_i, $0 \leqslant i < l$, where A_i consists of those elements in $\mathbb{F}_p^*$ whose logarithm to the base w is congruent to i modulo l. (So the A_i's can be thought of as the cosets of the subgroup $\langle w^l \rangle$ in $\mathbb{F}_p^*$.) Suppose r is a primitive pth root of unity in a finite extension $\mathbb{F}_q$ of $\mathbb{F}_l$. The ideal in the ring $\mathbb{F}_l[x]/(x^p - 1)$ generated by the polynomial $\prod_{a \in A_i}(x - r^a)$ is the lth power residue code $\mathfrak{A}_i$, and the ideal generated by the polynomial $(x-1)\prod_{a \in A_i}(x - r^a)$ is the lth power residue code $\bar{\mathfrak{A}}_i$

Suppose a is an arbitrary element of the set A_i. Clearly $aw^l \in A_i$ also, because $a \in A_i$ implies that the logarithm $\log_w(a)$ of a to the base w is congruent to i modulo l, and so we have $\log_w(aw^l) = \log_w(a) + l = i \bmod l$ also, so $aw^l \in A_i$. Thus set A_i is closed under multiplication by w^l, and so the roots of the polynomial $\prod_{a \in A_i}(x - r^a)$ are closed under exponentiation to the power w^l. Similarly the polynomials that make up any lth power residue code are closed under exponentiation by w^l. Thus again we can apply Theorem 6 here.

On p.412 of [4], the author discusses a known lth-root bound that applies to *some* of the codewords in a lth power residue code (namely those codewords in $\mathfrak{A}_i \setminus \bar{\mathfrak{A}}_i$,) and writes he "would like to see a lth-root bound hold for all codewords". The reason that the existing lth-root bound does not apply to all codewords lies in the possibility that minimal length codewords in $\mathfrak{A}_i$ have the root $r^0 = 1$. This is the precisely the situation that the second part of the proof of Theorem 6 deals with, and indeed that theorem applies to *all* polynomials in a lth power residue code.

Therefore, taking $\mathbb{G}$ to be the order-p subgroup of $\mathbb{F}_q^*$ generated by r, and assuming $B = (p-1)/l$ and $A = l$ are relatively prime, and taking $\beta = w^l$, where w is primitive in $\mathbb{F}_p$, then we can apply Theorem 6 to any polynomial in $\mathfrak{A}_i$, thus obtaining the following lower bound for the minimum distance of a lth power residue code, which is an improvement on the bound in [4] (for fixed l and as $p \to \infty$).

Theorem 7. *If primes p,l are primes with* $(p-1)/l$ *and* l *relatively prime, then* *the minimum distance of a* l*th-power residue code of length* p *satisfies*

$$d_{min}(\mathfrak{A}_i) \geqslant \sqrt[l]{\frac{\phi(p-1)}{l-1}}.$$

3.4 Polynomials constant on orbits

Here we characterize the structure of polynomials f constant on the orbits of size B, by which we mean $f(u) = f(v)$ if u, v are in the same orbit. Algorithms that evaluate such polynomials are type-2 distinguishers. The characterization given results in a lower bound on the number of nonzero coefficients in such polynomials.

Lemma 8. *Assume* $\mathbb{G}$ *is a multiplicative subgroup of prime order* p *of the finite field* $\mathbb{F}_q$*, and that* $p-1 = AB$ *for integers* $A, B \geqslant 2$ *with* $\gcd(A, B) = 1$*, and that integers* α, β *satisfy* $1 < \alpha, \beta < p$ *and* $\mathrm{ord}_p(\alpha) = A$ *and* $\mathrm{ord}_p(\beta) = B$*. Further suppose* $f \in \mathbb{F}_q[x]$ *is constant on the orbits of* $x \to x^\beta$ *on* $\mathbb{G}$*. Then* $f(x^\beta) - f(x)$ *is divisible by* $(x^p - 1)$*.*

Proof. Let $r \in \mathbb{G}$. Since f is constant on the orbits of $x \to x^\beta$, we have $f(r^\beta) = f(r)$. Hence $(x-r)$ is a factor of $f(x^\beta)-f(x)$ in $\mathbb{F}_q[x]$. Since this is true for an arbitrary r in $\mathbb{G}$, we have $\prod_{r\in\mathbb{G}}(x-r) = (x^p-1)$ divides $f(x^\beta)-f(x)$. □

The following theorem describes the structure of polynomials that are constant on the orbits.

Theorem 9. *Let* F *be a field, let* p *be a prime such that* $p - 1 = AB$ *for integers* $A, B \geqslant 2$*, let* $f \in F[x]$ *have* $\deg(f) < p$*, and let integers* α, β *satisfy* $1 < \alpha, \beta < p$ *and* $\mathrm{ord}_p(\alpha) = A, \mathrm{ord}_p(\beta) = B$*. Then* $f(x^\beta) - f(x)$ *is divisible by* $(x^p - 1)$ *if and only if there exist* $C, c_i \in F$ *such that*

$$f(x) = C + \sum_{a=0}^{A-1} c_a \sum_{b=0}^{B-1} x^{\beta^b\alpha^a}.$$

Proof. The "if" direction is trivial, so assume $f(x^\beta) - f(x)$ is divisible by $(x^p - 1)$. Further assume $f(x) = \sum_{i=0}^{p-1} c_i x^i$, and so $f(x^\beta) = \sum_{i=0}^{p-1} c_i x^{i\beta}$. Since $f(x^\beta) - f(x)$ is divisible by $(x^p - 1)$, by Lemma 5 we know that the polynomial $\sum_{i=0}^{p-1} c_i x^{(i\beta \bmod p)} - \sum_{i=0}^{p-1} c_i x^i$ must be the zero polynomial. Since $i\beta$ and $j\beta$ are in different residue classes modulo p if $0 \leqslant i < j < p$, we see that for each integer $n \in [0, p-1]$ we have $c_{n^*} - c_n = 0$, where the integer $n^* \in [0, p-1]$ satisfies $\beta n^* \equiv n \bmod p$, that is $n^* = (n\beta^{-1} \bmod p)$. In other words, $c_{n\beta^{-1}} = c_n$, where we understand that subscripts are taken modulo p. Similarly $c_{n\beta^{-2}} = c_{n\beta^{-1}}$, so $c_{n\beta^{-2}} = c_n$, and so on. Hence any two coefficients c_u and c_v will be equal if u and v are in the same orbit of $\mathbb{F}_p$ under the action of multiplication by β^{-1}. But since $1 < \beta < p$, multiplication by β is a bijection on $\mathbb{F}_p$ and so the orbits of multiplication by β^{-1} are the same as the orbits of multiplication by β. This gives the form of the polynomial in the theorem statement. □

This last result shows that a non-constant polynomial that is constant on the orbits (of $x \to x^{\beta}$) must have at least B nonzero coefficients. Thus, the algorithm D of Section 2.2 , if constructed by evaluating such a polynomial using Horner's rule (or other 'one-coefficient-at-a-time' based approaches) would require at least B operations, and so the complexity of Algorithm 2 from that section would be at least $\Theta(\sqrt{A}B)$ which is not an improvement on a square root algorithm.

4 Acknowledgements

The author would like to thank the reviewers of earlier versions of this paper, in particular Alfred Menezes, Stephen Galbraith, and Edlyn Teske. Thanks also to N. Hynes, C. Dawe, and I. Payne.

References

[1] Alexandra Boldyreva. Efficient threshold signature, multisignature and blind signature schemes based on the gap-Diffie-Hellman-group signature scheme. In *Proceedings of PKC 2003, volume 2567 of LNCS*, pages 31–46. Springer-Verlag, 2003.

[2] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[3] Dan Boneh and Richard Lipton. Searching for elements in black box fields and applications. In *In Advances in Cryptology-Crypto96, LNCS1109*, pages 283–297. Springer-Verlag, 1996.

[4] P. Charters. Generalizing binary quadratic residue codes to higher power residues over larger fields. *Finite Fields and Their Applications*, 15(3):404–413, June 2009.

[5] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. *Advances in Cryptology — EUROCRYPT 2006, LNCS*, 4004:1–11, 2006.

[6] R. P. Gallant D. R. L. Brown. The static Diffie-Hellman problem. *eprint.iacr.org/2004/306.ps*, 2004.

[7] Eike Kiltz and Arne Winterhof. Polynomial interpolation of cryptographic functions related to Diffie-Hellman and discrete logarithm problem. *Discrete Appl. Math.*, 154(2):326–336, 2006.

[8] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*, volume 16 of *North-Holland Mathematical Library.* North-Holland, 1977.

[9] Ueli M. Maurer and Stefan Wolf. The Diffie–Hellman protocol. *Des. Codes Cryptography*, 19(2-3):147–171, 2000.

[10] Wilfried Meidl and Arne Winterhof. A polynomial representation of the Diffie-Hellman mapping. *Appl. Algebra Eng. Commun. Comput.*, 13(4):313–318, 2002.

[11] Alfred J. Menezes and Neal Koblitz. Another look at non-standard discrete-log and Diffie-Hellman problems. *Journal of Mathematical Cryptology*, 2(4):311–326, 2008.

[12] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography.* CRC Press, 1997.

[13] M. J. Wiener P. C. van Oorschot. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12:1–28, 1999.

[14] D. Shanks. Class number, a theory of factorization and genera. *Proc. Symp. Pure Math.*, 20:415–440, 1971.

[15] V. Shoup. Lower bounds for discrete logarithms and related problems. *Advances in Cryptology — EUROCRYPT 1997, LNCS*, 1233:256–266, 1997.

[16] Douglas R. Stinson. *Cryptography: theory and practice.* CRC Press, 1995.

A Pollard-rho approach

The algorithms in Section 2 used a variant of Shanks' Baby-Step-Giant-Step algorithm [14], which, though simple to describe, requires much storage and is not linearly parallelizable. We outline how the algorithm in Section 2.1 could be implemented using $2m$ parallel processors and a central processor, using the ideas of [13], which addresses these deficiencies. The algorithm in Section 2.2 can also be so modified.

Given a generator G of $\mathbb{G}$, an element $W \in \mathbb{G}$, where G has order $p = AB+1$, and $W = G^w$ for some $w \in [0, p-1]$.

Let α, β be elements of $\mathbb{F}_p^*$ with $\mathrm{ord}_p(\alpha)$, the multiplicative order of α in $\mathbb{F}_p$, equal to A, and $\mathrm{ord}_p(\beta) = B$, and $\gcd(A, B) = 1$, so that α, β generate $\mathbb{F}_p^*$. Then $w = \alpha^{w_1}\beta^{w_2}(\mathrm{mod} p)$ for some integers w_1, w_2. We find w_1, w_2, and hence w.

Steps 1,2, and 3 are as before. Steps 4 and 5 are replaced by the following.

4. Set $G' = G^{\alpha^u}$, $W' = W^{\alpha^v}$. Since $D(G') = D(W') = 1$, both G' and W' are in the orbit distinguished by D.

   (a) On processors $1 \dots m$, start the processor by picking a random integer $r \in [0 \dots B-1]$ and forming the tuple (G'^{β^r}, r, G').

   (b) On processors $m+1 \dots 2m$, start the processor by picking a random integer $r \in [0 \dots B-1]$ and forming the tuple (W'^{β^r}, r, W').

(c) On all processors, perform the following walk: Given tuple $(P, k, *)$, compute a pseudo-random number r determined (only) from a canonical representation of the element P, and replace tuple $(P, k, *)$ by the tuple $(P^{\beta^r}, (r + k) \bmod B, *)$.

(d) If the new element P^{β^r} is distinguished (for example, it contains a specific bit pattern in its canonical representation) then send this tuple to the central processor, and restart this processor as above.

(e) After approximately $\sqrt{B}$ walk "steps" in total , we expect the central processor will receive different tuples containing the same first coordinate. When this happens, with probability $1/2$ one tuple was sent from the first half of the processors, and the other tuple was sent from the other half. So we have tuples (M, m, G') and (M, n, W') with $G'^{\beta^m} = W'^{\beta^n}$, which gives $G^{\alpha^u \beta^m} = W^{\alpha^v \beta^n}$.

5. As $G^{\alpha^u \beta^m} = W^{\alpha^v \beta^n}$, we have the discrete logarithm of W to the base G is $\frac{\alpha^u \beta^m}{\alpha^v \beta^n} \bmod p$.