.

Improvements to the Gaudry-Schost Algorithm for Multidimensional discrete logarithm problems and Applications

by

Raminder Singh Ruprai

Thesis submitted to the University of London

for the degree of Doctor of Philosophy

Department of Mathematics

Royal Holloway University of London

2010

Declaration

These doctoral studies were conducted under the supervision of Professor Steven D. Galbraith.

The work presented in this thesis are the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Department of Mathematics as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Raminder Singh Ruprai

2^{nd} September 2009.

Abstract

The discrete logarithm problem (DLP) is a problem used throughout cryptography and information security as the bedrock of the security of many systems. The DLP applies in any group but is only considered a hard problem in certain groups. We focus on the DLP in these groups and particularly the DLP in an interval that is smaller than the order of the group.

The Pollard kangaroo algorithm solves the DLP in an interval of size N with heuristic expected running time approximately $2\sqrt{N}$ group operations. It is well-known that the Pollard rho method can be sped-up by using equivalence classes, but such ideas have not been used for the DLP in an interval. Pollard very recently showed how to solve the DLP in an interval with heuristic expected running time of less than $1.71\sqrt{N}$ group operations, using just one inversion in the group. Our two main results in this area are to give algorithms to solve the DLP in an interval of size N with heuristic expected running time of less than $1.72\sqrt{N}$ group operations for a generic group and $1.47\sqrt{N}$ group operations for groups with fast inversion.

We also present an improvement to the Gaudry-Schost low-memory algorithm for solving the 2-dimensional DLP and extend this improvement to the general

multidimensional DLP. The main tool of the algorithm is a multidimensional pseudorandom walk which we analyse thoroughly in the 1, 2 and 3 dimensional cases as well as giving some discussion for higher dimensions.

Acknowledgements

First and foremost I would like to thank my supervisor Professor Steven D. Galbraith for his support, encouragement, disappointment and enthusiasm with all the research we carried out.

I want to thank my family for taking the 'flak' when my research was slow and for being there to remind me that I would reach the end, successfully. In addition I especially want to thank my partner and companion, Mariela, who was always a shoulder to lean on during those days of fruitless research and had faith in me that by persevering, good research would follow.

I would like to thank Royal Holloway for the Thomas Holloway Scholarship which allowed me to complete this thesis without the financial worry.

Last but not least I would like to thank all the researchers and academics listed in the Bibliography especially John Pollard, Perrick Gaudry and Éric Schost for without their work, the progress I have made in this area could not have happened.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The discrete logarithm problem (DLP) is a problem used throughout cryptography and information security as the bedrock of the security of many systems. The DLP can be found in security standards such as the Digital Signature Standard [23]. It also forms the basis of the Elgamal cryptosystem [9] and the Diffie-Hellman key exchange protocol [8]. The hardness of solving this problem is what makes cryptosystems which use the DLP secure. However there are algorithms for solving the DLP. As these algorithms improve, meaning that their running times decrease, the security of these systems fall. In this thesis we look at a number of algorithms for solving variants of the DLP. In particular we focus on the DLP in an interval in Chapter 3. The main difference between this problem and the standard DLP is that the interval of possible solutions is smaller than the order of the group. This problem arises naturally in a number of contexts, for example the DLP with c-bit exponents (c-DLSE) [20, 33, 47], counting points on curves or abelian varieties over finite fields [19], the analysis of the strong Diffie-Hellman

problem [5, 24], and side-channel or small subgroup attacks [21, 29].

- In Chapter 2 we look at the standard DLP. We briefly discuss the problem in special groups such as prime fields and the subexponential algorithms that work in those special cases. However the main part of this chapter will be to describe in detail two algorithms to solve the DLP in a generic group namely Shanks' Baby-Step Giant-Step and Pollard's Rho. The running times of these algorithms are similar but the main difference is that Pollard's Rho is a low memory algorithm and that is what we focus on throughout the thesis.

- In Chapter 3 we define the DLP in an interval which is a variant of the DLP that we will spend the most time on. Here we describe the current algorithms for solving this problem namely Pollard's kangaroo algorithm and the method of van Oorchost and Wiener which parallelises the kangaroo algorithm. Finally we finish off this chapter with an explanation of the Tame-Wild Birthday Paradox which is required in the following chapter.

- In Chapter 4 we introduce the Gaudry-Schost algorithm together with its different form of analysis that utilises the Tame-Wild Birthday Paradox. These algorithms are all well-known in the field. At this point we build on the literature by

  - Improving the Gaudry-Schost algorithm,

  - Presenting all the details of Pollard's own improvement of his kangaroo algorithm,

– And making further improvements to the Gaudry-Schost algorithm following new insights of Pollard.

Finally we compare the running times of all the algorithms up to this point as well as giving experimental data to support them. Together with Pollard we are in the process of writing a research paper on his improved kangaroo algorithm and our improved Gaudry-Schost algorithm [13].

- In Chapter 5 we consider the DLP in an interval in groups which have fast inversion such as the group of points on an elliptic curve. Then using equivalence classes we make further improvements to the Gaudry-Schost algorithm to obtain the fastest algorithm for solving the DLP in an interval with groups with fast inversion. We again give a comparison of all the algorithms as well as experimental results of the algorithms presented in this chapter. Our research paper on this topic [14] was accepted to the Indocrypt 2009 conference in New Delhi, India but has since been withdrawn and resubmitted to the Public Key Cryptography 2010 conference in Paris, France.

- Finally in Chapter 6 we consider the analogous multidimensional problem of the DLP in an interval. First we motivate the multidimensional DLP by providing a number of examples in cryptography where the multidimensional problem appears and therefore showing that any improvements to the current algorithm are useful. In addition we give an example of where the multidimensional DLP arises in the process of counting points on curves. We present the general Gaudry-Schost algorithm for solving the multidimensional DLP. Then we describe in detail our improvement to this

algorithm as well as give a thorough analysis of the bounds for the number of bad steps that the algorithm is likely to produce. Our research paper on the improved algorithm and its analysis [15] was published in the proceedings of the 12th IMA International Conference on Cryptography and Coding in Cirencester, UK.

Chapter 2

Computing Discrete Logarithms in Generic Groups

In this chapter we define the discrete logarithm problem and the elliptic curve discrete logarithm problem which form the motivation for this thesis. We present and discuss the two main algorithms for solving these problems in the general case along with their running times. The first is the Baby-Step Giant-Step algorithm which has the smallest running time but a large memory requirement. This is a deterministic algorithm and as such has a probability of success equal to 1. Then we look at the Pollard rho algorithm which has a similar running time but very low memory requirement. This is a probabilistic algorithm and has a possibility of failure (not solving the DLP) although this is negligible for large groups.

2.1 Elliptic Curve Discrete Logarithm Problem

Throughout this thesis we will use elliptic curve notation to explain the many algorithms to solve the discrete logarithm problem in a generic abelian group. Traditionally the group law in an abelian group is written multiplicatively however on an elliptic curves it is usually written additively. Therefore we will use additive notation as introduced in the next definition.

Definition 2.1.1. Let G be a group with a binary operator written additively. Let $P \in G$ then

$$[k]P = \underbrace{P + P + \cdots + P}_{k \text{ times}}.$$

Using the notation above the DLP is as follows.

Definition 2.1.2. Let G be an abelian group and let $P, Q \in G$, then the Discrete Logarithm Problem is to find the smallest positive integer n such that

$$Q = [n]P. \tag{2.1}$$

The Elliptic Curve Discrete Logarithm Problem (ECDLP) is the same as the above except that the group G is $E(\mathbb{F}_q)[r]$ which is defined in the next definition.

Definition 2.1.3. Let E be an elliptic curve over $\mathbb{F}_q$ and denote by $E(\mathbb{F}_q)$ the set of points on the curve with coordinates in $\mathbb{F}_q$. Further, let $E(\mathbb{F}_q)[r]$ be the set of r-torsion points on the elliptic curve i.e.

$$E(\mathbb{F}_q)[r] = \{P \in E(\mathbb{F}_q) \mid [r]P = \mathcal{O}\}.$$

The DLP is regarded as a 'hard' problem meaning that there is no known polynomial time algorithm for solving it in a generic group. There are many cryptosystems such as Elgamal [9] that use the hardness of the DLP for their security. Elliptic curves are specifically used in Elliptic Curve Integration Encryption Scheme (ECIES) [4]. The DLP is also used in the Diffie-Hellman key exchange [8], the Digital Signature Standard [23] and many other signature schemes. Originally the group used in many of the cryptographic applications was the multiplicative group of a finite field $\mathbb{F}_q$ where q is a prime power. In this specific group the DLP can be solved in subexponential running time. To present the running time of an algorithm which terminates in subexponential time we use the following equation

$$L_r[v;c] = e^{(c+o(1))(\log r)^v(\log\log r)^{1-v}}. \tag{2.2}$$

The first subexponential algorithm for solving the DLP in a prime field (i.e. $\mathbb{F}_p$ where p is prime) is the Index calculus algorithm. A basic description of this algorithm can be found in Stinson [43] and Ruprai [37]. Using equation (2.2) the running time for this algorithm is $L_r[1/2;1]$. Adleman used a very similar approach on the finite field $\mathbb{F}_q$ where q is a power of 2, and achieved the same running time. Coppersmith improved on Adleman's result to obtain an algorithm with running time $L_r[1/3;c]$ for some small constant c. This method was later extended to finite fields of many forms by using number field and function field sieves as shown by Gordon [22], Adleman and Demarris [1], Schirokauer et. al. [38], Joux and Lercier [25] and more recently Joux, Lercier, Smart and Vercauteren [26]. The running time remains $L_r[1/3;c]$ where the constant c varies depending on the particular finite field.

Although we will be using elliptic curve notation throughout the thesis, many of the applications will not be restrained to elliptic curves so we will consider algorithms for solving the DLP as opposed to just the ECDLP.

2.2 Shanks' Baby-Step Giant-Step algorithm

Shanks' Baby-Step Giant-Step (BSGS) algorithm [40, 42] solves the DLP in running time $O(\sqrt{r})$ group operations. We give brief details as to how the algorithm proceeds. The discrete logarithm has bounds $0 < n < r$, so let $m = \lceil \sqrt{r} \rceil$. Then we can write the discrete logarithm as $n = im + j$ for $0 \leq i, j < m$ and so the DLP becomes

$$\begin{aligned} Q =& [im + j]P \\ Q - [j]P =& [im]P. \end{aligned}$$

We precompute and store $(j, Q - [j]P)$ for $0 \leq j < m$. These are known as the 'baby-steps'. Therefore we are storing approximately $\sqrt{r}$ pairs which have to be sorted by their second component i.e. $Q - [j]P$ as we will be searching through this list a number of times. We then evaluate $[im]P$ ('giant-steps') for $0 \leq i < m$ and for each i we check whether this is equal to the second component of one of the pairs we have stored in the easily accessible data structure. When we have a match the first component of the pair gives us the value of j and we have the value of i from the corresponding giant-step so we can easily calculate $n = im + j$. Algorithm 1 presents the BSGS algorithm using the notation as laid out here.

Algorithm 1 Shanks' Baby-Step Giant-Step algorithm

INPUT: $P, Q \in G$, $r = \#G$
OUTPUT: An integer n such that $Q = [n]P$

1: $m := \lceil \sqrt{r} \rceil$
2: Let T be a hash table to store 2-tuples and sorted by the element of G
3: $x := Q$
4: **for** $j \in \{0 \dots m-1\}$ **do**
5: Insert (j, x) into the correct place in T
6: Update $x := x - P$
7: **end for**
8:
9: $y := 0$
10: **for** $i \in \{0 \dots m-1\}$ **do**
11: Search for $(j*, y)$ in T for some $j*$
12: **if** $(j*, [im]P) \in T$ for some $j*$ **then**
13: **return** $im + j*$
14: **end if**
15: Update $y := y + [m]P$
16: **end for**

If we neglect the searching time of the the giant-steps, the running time of Shanks' algorithm is $\sqrt{r}$ group operations for the precomputation phase and a further maximum $\sqrt{r}$ group operations for the giant-steps. Therefore the running time is $O(\sqrt{r})$. However the storage requirement of this algorithm is also non-trivial. As we have to store all the baby-steps the storage requirement is $O(\sqrt{r})$ group elements. For large groups, Shanks' algorithm is unusable in practice as the storage requirement is too large. In the next section we consider an algorithm which has the same running time but is low memory and therefore more suitable for implementation purposes.

2.3 Pollard's Rho algorithm

Pollard's rho algorithm was first introduced by Pollard [34] to solve the DLP and has since been improved by Brent [3] and Teske [44]. The advantage of the rho algorithm over the BSGS algorithm is that it requires very low memory while still having a running time of $O(\sqrt{r})$ group operations. However unlike the BSGS algorithm, the rho algorithm is randomised, so the running time is only an expected value. We describe the main parts of the algorithm but we need to first revise some theory on pseudorandom walks.

2.3.1 Pseudorandom walks

The goal of a pseudorandom walk is to mimic (as much as possible) selecting elements from G uniformly at random. Of course we can never obtain a truly random walk but we believe that we can get close enough to it. The pseudorandom walk needs to be deterministic, meaning that there is no random input when deciding on the next step of the walk. In this way two different walks which land on the same point will continue along the same path as both walks are determining their next steps from the same well defined function. This will enable us to solve the discrete logarithm problem as follows; First we partition G into n_s distinct sets, denoted by S_i, as follows

$$G = S_0 \cup S_1 \cup \ldots \cup S_{n_s-1}.$$

Using this partitioning of G we define a selection function which takes as input an element of G and outputs the index, $\{0, \ldots, n_s - 1\}$, of the partition it belongs

to.

Definition 2.3.1. A selection function $S : G \to \{0, 1, \ldots, n_s - 1\}$ maps elements of G to the integers modulo n_s so the partitions of G can be written as $S_i = \{g \in G \mid S(g) = i\}$.

The design of a selection function is only briefly mentioned in the literature as a hash function (Pollard, [35]) so we now give an example related to elliptic curves.

Example 2.3.2. Let G be the group of r-torsion points on an elliptic curve over a finite field i.e. $G = E(\mathbb{F}_q)[r]$ where q is a prime power. Then the selection function S maps the points in G to the integers modulo some n_s. Let $P = (x_P, y_P) \in G$ where $x_P, y_P \in \mathbb{F}_q$. As points on an elliptic curve can be defined by their x_P co-ordinate and a sign, the selection function need only look at the first co-ordinate of P. As $q = p^t$ for some prime p and natural number t, elements of $\mathbb{F}_q$ can be represented as polynomials of degree strictly less than t. So

$$x_P = c_{t-1}x^{t-1} + c_{t-2}x^{t-2} + \ldots + c_1 x + c_0$$

where $c_i \in \mathbb{Z}_p$ for $i \in \{0, \ldots, t-1\}$. We can write each c_i as a binary string $((c_i)_2)$ and concatenate to have a unique binary string for each element of $\mathbb{F}_q$. We convert the binary string to an integer modulo n_s and output as required. Therefore the selection function is given by

$$S(x_P, y_P) = (c_{t-1})_2 \| (c_{t-2})_2 \| \ldots \| (c_1)_2 \| (c_0)_2 \mod n_s.$$

In fact, elements of $\mathbb{F}_q$ are stored on a computer as t-tuples of binary strings (the coefficients). Therefore the computing cost of the concatenation and the

subsequent binary to integer modulo n_s conversion is effectively free.

Assumption 2.3.3. If the group order of G is significantly large then the output of the selection function (as described in Example 2.3.2) is sufficiently random that it partitions the elements of G evenly into n_s distinct sets.

Definition 2.3.4. A pseudorandom walk on a group G is a sequence of elements of G, i.e. $R_1, R_2, \ldots, R_n$ such that the next element of the sequence is determined by function with no random element to the input.

For each partition of G, S_i, we associate a 'step size' or increment, t_i, in the exponent of the current element of the pseudorandom walk. These step sizes can be predetermined or can be a function of the current exponent such as doubling the exponent. We call the R_is the footprints of the walk. For example the next footprint of the pseudorandom walk could be calculated as follows

$$R_{i+1} = f(R_i) = R_i + [t_{S(R_i)}]P.$$

2.3.2 The idea of Pollard Rho

We will use the notation laid out in Definition 2.1.2 to describe the rho algorithm. The idea of the Pollard rho algorithm is to find integers (a_i, b_i) and (a_j, b_j) such that

$$[a_i]P + [b_i]Q = [a_j]P + [b_j]Q.$$

We call this a collision. Pollard uses a pseudorandom walk to find a collision and we then solve the DLP as

$$n = (a_i - a_j)(b_j - b_i)^{-1} \pmod{r}$$

assuming that $(b_j - b_i)$ is invertible modulo r. Now we look at the pseudorandom walk that Pollard uses. We take $n_s = 3$ so we are partitioning the group as $G = S_0 \cup S_1 \cup S_2$. Let R_i be the elements that the pseudorandom walk visits and we start by taking $R_1 = P$ then the pseudorandom walk is as follows

Pseudorandom Walk - 1D 1 (Rho)**.**

$$R_{i+1} = f(R_i) = \begin{cases} R_i + P & \text{if } S(R_i) = 0 \\ [2]\, R_i & \text{if } S(R_i) = 1 \\ R_i + Q & \text{if } S(R_i) = 2. \end{cases}$$

Each step only requires one group operation and one call to the selection function which can be made effectively free. In addition to each group operation we need to keep track of the decomposition of R_i. When $i = 1$ we set $a_1 = 1$ and $b_1 = 0$. We update these exponents as follows

$$(a_{i+1}, b_{i+1}) = \begin{cases} (a_i + 1, b_i) & \text{if } S(R_i) = 0 \\ (2a_i, 2b_i) & \text{if } S(R_i) = 1 \\ (a_i, b_i + 1) & \text{if } S(R_i) = 2, \end{cases}$$

A final issue remains as to how a collision, $R_i = R_j$ where $i \neq j$, is found. As G is finite the pseudorandom walk will eventually cycle which will enable us to

find such a collision. One approach could be to store all the R_i and for each new R_j we can check whether that group element has already been stored. In fact this naive method will store $\sqrt{\pi r/2}$ group elements which is more than the BSGS algorithm to solve the DLP. Instead we can view the pseudorandom walk as having a 'tail' (the part of the walk which is not cyclic) and a 'head' (the cyclic part of the walk) which one can say forms a 'rho' shape, hence the name of the algorithm. Let l_h and l_t be the lengths of the head and tail respectively then the first collision is

$$R_{l_t+l_h} = R_{l_t}.$$

Floyd's cycle finding method [10], which also appears in Knuth [27], compares R_i and R_{2i} for a collision. So in practice we have two pseudorandom walks which both start at the same point $R_1 = P$. The first walks one step at a time and the second walks two steps at a time. This way we only ever have to store two group elements with their associated exponents.

Lemma 2.3.5. *Let the notation be as above then we have the following:-*

1. $R_i = R_{2i} \Leftrightarrow l_h | i$ *and* $i \geq l_t$.
2. *There is some* $l_t \leq i < l_t + l_h$ *such that* $R_i = R_{2i}$.

Proof. We prove each part of the lemma separately

1. The difference between the indices i and $2i$ must be a multiple of the length of the head therefore $2i - i = cl_h$ where c is a positive integer. So $l_h | i$ and as a collision can only occur after the tail it follows that $i \geq l_t$.
2. This statement follows as there is a multiple of l_h between l_t and $l_t + l_h$.

□

Floyd's cycle finding method has since been improved by Brent [3] who claims that, on average, his cycle finding method is 30% quicker than Floyd's and also speeds up the Pollard rho algorithm by 24%. Sedgewick, Szymanski and Yao [39] showed that by storing a few more of the elements visited on the pseudorandom walk, a collision can be found around three times faster. Although the storage in this case is greater than $O(1)$ group elements it is far less than the storage requirement of BSGS. More recently Cheon, Hong and Kim [6] have shown that by not fully calculating every step of the pseudorandom walk Pollard rho can be made up to 10 times faster when the group is a prime field.

2.3.3 The basic algorithm

We now present the basic Pollard rho algorithm. For simplicity we will take the steps of the pseudorandom walk to be just an extended version of the function f defined in Definition 2.3.4, which takes as input (R_i, a_i, b_i) and returns the triple $(R_{i+1}, a_{i+1}, b_{i+1})$. Although the algorithm shows a possibility of failure in line 20 the likelihood of this is negligible in large groups.

2.3.4 Running Time Analysis

It remains an open problem to give a rigorous running time analysis of the Pollard rho algorithm as we require a heuristic assumption.

Theorem 2.3.6 (Birthday Paradox)**.** *Given a set $\mathcal{A}$ of r elements, if elements are sampled uniformly at random from $\mathcal{A}$ then the expected number of samples to*

Algorithm 2 The Pollard rho algorithm

INPUT: $P, Q \in G$

OUTPUT: An integer n such that $Q = [n]P$, or $\perp$

```
Choose randomly the function S as explained above
function f(R, a, b)
    if S(R) = 0 mod 3 then
        f(R, a, b) := (R + P, a + 1, b)
    else if S(R) = 1 mod 3 then
        f(R, a, b) := ([2]R, 2a, 2b)
    else if S(R) = 2 mod 3 then
        f(R, a, b) := (R + Q, a, b + 1)
    end if
    return f
end function

R_1 := P, a_1 := 1, b_1 := 0
(R_2, a_2, b_2) := f(R_1, a_1, b_1)
while (R_1 ≠ R_2) do
    (R_1, a_1, b_1) := f(R_1, a_1, b_1)
    (R_2, a_2, b_2) := f(f(R_2, a_2, b_2))
end while
if b_1 ≡ b_2 mod r then
    return ⊥
else
    return (a_2 − a_1)(b_1 − b_2)^{-1} mod r
end if
```

be taken before some element is sampled twice is less than

$$\sqrt{\frac{\pi r}{2}} + 2 \approx \sqrt{\frac{\pi r}{2}}.$$

The Birthday Paradox is stated in many references but the basic probabilistic argument can be found in Menezes, Vanstone and van Oorschot [30] whilst a full proof can be found in Galbraith [11].

Heuristic 2.3.7. As r becomes large then

1. the Pseudorandom walk 1 behaves like a random walk.

2. there exists a small $\epsilon > 0$ such that the values of l_t and l_h are independent with expected value $(1+\epsilon)\sqrt{\pi r/8}$.

Part 2 of Heuristic 2.3.7 follows from the Birthday Paradox, as the first collision is expected in $l_t + l_h = \sqrt{\pi r/2}$ steps when the walk is random. Smart [41] shows that if the pseudorandom walk picks elements uniformly at random from G then $\epsilon = 0$ but in practice the pseudorandom walk can never be truly random although we can get close to it. Teske [44] shows that the original Pollard rho with $n_s = 3$ is not optimal. In practice one should take $n_s \geq 20$ and Pseudorandom walk 1 will have n_s different steps which are randomly chosen before the algorithm begins. In such a setting part 2 of Heuristic 2.3.7 seems to be true for $\epsilon < 0.04$.

Theorem 2.3.8. *Let the notation be as above and assume Heuristic 2.3.7 then the Pollard rho algorithm has expected running time of* $3(1+\epsilon)\sqrt{\pi r/2}$ *when using Floyd's cycle finding method.*

Proof. By the Birthday Paradox (Theorem 2.3.6) we expect to see a collision within $(1+\epsilon)\sqrt{\pi r/2}$ iterations of the while loop in Algorithm 2. When using Floyd's cycle finding method, we make 3 functions calls per iteration of the loop so the result follows. □

Chapter 3

Current Methods for Computing Discrete Logarithms in Intervals

In this chapter and the following, we consider a slight variation of the discrete logarithm problem which we call the discrete logarithm problem in an interval.

Definition 3.0.9. Let G be a group of order r. Let $P_1, Q \in G$ and $N_1 \in \mathbb{N}$ be given. Then the discrete logarithm problem in an interval is to find an integer $n_1 \in [-N_1, N_1]$ such that

$$Q = [n_1]P_1. \tag{3.1}$$

N.B. Let the size of the entire search space (total number of possible values of n_1) be given by $N = 2N_1 + 1$. Also the interval of exponents $[-N_1, N_1]$ refers to the interval of integers $-N_1 \leq i \leq N_1$.

Without loss of generality we have restricted Definition 3.0.9 to intervals centred around 0. We can do this because if we are given a DLP in an interval, say $[x, y]$,

we can shift the problem instance so that the interval is centred around 0. We do this by taking the middle of the interval, so in this case $(x + y)/2$, and shift our problem instance thus,

$$Q' = Q - [(x + y)/2]P_1.$$

Therefore our new problem instance Q' will have a discrete logarithm in the interval

$$\left[-\frac{y-x}{2}, \frac{y-x}{2}\right].$$

In this chapter we present the current best low-memory algorithm for solving this problem namely Pollard's kangaroo algorithm. Then we present a parallel version of this algorithm by van Oorschot and Wiener. Finally we finish off this chapter with an explanation of the Tame-Wild Birthday Paradox which is required in Chapter 4.

3.1 Pollard's kangaroo algorithm

Pollard's kangaroo or 'lambda' algorithm [34] specifically tackles the DLP in an interval. We will use the notation as described in Definition 3.0.9. The main idea of this algorithm is that we have two kangaroos or pseudorandom walks; a tame kangaroo which starts at the element at the right hand end of the interval of exponents and a wild kangaroo which starts at Q. The tame kangaroo takes a prescribed number of steps and we store the final footprint together with its discrete logarithm. This is known as the 'trap'. Then the wild kangaroo also takes a prescribed number of steps and we hope that one of the steps lands on a footprint of the tame kangaroo. In which case the wild kangaroo will continue on the same path as the tame kangaroo and land in the trap and so we can solve for the DLP.

3.1.1 Algorithm in detail

We have seen in Section 2.3 how to construct a pseudorandom walk and also the walk used in Pollard's rho algorithm. The pseudorandom walk here is somewhat different. In practice the number of partitions of G, n_s, is taken as 20 or 32. Also each of the different steps of the pseudorandom walk is smaller than those in Pollard's rho algorithm. Let x_i be the footprints of the pseudorandom walk and let a_i be the exponent of the element visited at stage i.

Pseudorandom Walk - 1D 2 (kangaroo)**.** The pseudorandom walk proceeds as follows,

- We have a selection function S as described in Section 2.3.1 which takes

as input the current footprint of the pseudorandom walk and outputs the index of the partition of G which that element belongs to.

- For $0 \leq j < n_s$ we choose random step sizes $1 \leq u_j \leq 2c\sqrt{N}$ (where the constant c is determined later) and then precompute $[u_j]P_1$. Pollard [34, 35] used powers of 2 as the step sizes i.e. $u_i = 2^i$. In practice these work well but Pollard does not claim that this is the best choice.

- We define the mean step size

$$m = \frac{1}{n_s} \sum_{j=0}^{n_s-1} u_j.$$

  We want $m \approx c\sqrt{N}$ and one can choose the u_j's in such a way that this occurs as shown by Pollard.

- The pseudorandom walk is as follows

$$x_{i+1} = f(x_i) = x_i + [u_{S(x_i)}]P_1.$$

  We update the exponent as well i.e. $a_{i+1} = a_i + u_{S(x_i)}$.

The tame kangaroo starts at the element $x_1 = [N_1]P_1$ and continues for $T = tm$ steps. Let $t \in \mathbb{R}_{\geq 1}$ therefore the total number of tame kangaroo steps is some multiple of the mean step size. The value of t is decided dependant on what probability of success of the algorithm we require (see Section 3.1.2). Then we store the final footprint or 'trap' along with its discrete logarithm, (x_T, a_T). The wild kangaroo starts at $y_1 = Q$ and continues until it lands on the trap or has gone past it. The discrete logarithm of the trap, a_T, is also the distance from

the middle of the interval to the trap, so the wild kangaroo will have passed the trap (for all problem instances) after it has traveled a distance greater than $N_1 + a_T + 1$. Let b_i be the exponent of the wild kangaroo at stage i. At each step we test whether $y_i = x_T$. If so we have

$$Q + [b_i]P_1 = y_i = x_T = [a_T]P_1$$

and so $n_1 = (a_T - b_i) \mod r$ solving the DLP in an interval. We present Pollard's kangaroo in Algorithm 3.

Algorithm 3 The Pollard kangaroo algorithm

INPUT: $P_1, Q \in G$ and $N_1 \in \mathbb{N}$
OUTPUT: An integer n_1 such that $Q = [n_1]P_1$, or $\perp$

Choose n_s to be the number of partitions of G
Choose t dependent on the probability of success required
Choose randomly the function S as explained in Section 2.3.1
function $f(R, a)$
    $f(R, a) := (R + [u_{S(R)}]P_1, a + u_{S(R)})$
    return f
end function

$x := [N_1]P, a := N_1$
for $i \in \{1, \ldots, \lceil (1+\epsilon)ct\sqrt{N} \rceil\}$ **do**
    $(x, a) := f(x, a)$
end for
$y := Q, b := 0$
while $(y \neq x)$ **do**
    if $b > N_1 + a + 1$ **then**
        return $\perp$
    end if
    $(y, b) := f(y, b)$
end while
return $(a - b) \mod r$

3.1.2 Running Time Analysis

Unlike Pollard's rho algorithm the Birthday Paradox is not utilised to analyse the running time of the kangaroo algorithm. Instead we use a mean step size approach and optimise the running time by choosing the constant c depending on whether we are optimising for the worst or average cases. However there is a tradeoff between the running time and probability of success as we will see.

We have set up the algorithm such that the tame walk takes tm steps which are of size m on average. To recap, when we discuss the size of a step we are referring to the increment in the exponent. Therefore the tame kangaroo covers a distance tm^2 from the right hand edge of the interval of possible exponents. So we expect that in each subinterval of length m, after the exponent N_1, there is one tame footprint. The wild kangaroo has to use the same deterministic pseudorandom walk so has the same mean step size m. Once the wild kangaroo passes the exponent N_1, in every subinterval of size m we also expect it to have one footprint. Therefore the probability that the wild kangaroo lands on the footprint of the tame walk is $1/m$. We deduce that the probability of no collision is

$$\left(1 - \frac{1}{m}\right)^{tm} \leq \left(e^{-1/m}\right)^{tm} = e^{-t}.$$

So the probability of a collision and for solving the DLP in an interval is approximately $1 - e^{-t}$. This is the probability of success. We have assumed that the tame and wild steps are uniformly spread out in the interval $[N_1, N_1 + tm^2)$ so we are assuming the following heuristic holds.

Heuristic 3.1.1. Using the pseudorandom walk described above there is a small

$\epsilon \geq 0$ such that if the tame kangaroo takes $(1+\epsilon)tm$ steps and the wild kangaroo takes $(1+\epsilon)tm$ in the same region then the probability of success is $1 - e^{-t}$.

For example when $t = 1, 2, 3, 4$ the probability of success is $0.63, 0.86, 0.95$ and 0.98 respectively. Using Heuristic 3.1.1 we can optimise the running time of the kangaroo algorithm for both the worst and average cases.

Theorem 3.1.2. *Using Pollard's kangaroo algorithm to solve the DLP in an interval of size N and assuming Heuristic 3.1.1 holds,*

1. *If we optimise for the worst case then $c = 1/\sqrt{2}$ and the average case expected running time is $(2+t)(1+\epsilon)\sqrt{N/2t}$ group operations and the worst case expected running time is $2(1+\epsilon)\sqrt{2tN}$ group operations.*

2. *If we optimise for the average case then $c_1 = 1/2$ and the average case expected running time is $2(1+\epsilon)\sqrt{tN}$ group operations and the worst case expected running time is $3(1+\epsilon)\sqrt{tN}$ group operations.*

Proof. 1. In the worst case Q lies on the left hand edge of the interval of exponents. Therefore for the wild kangaroo to 'leap' across the interval it must take $(1+\epsilon)N/m$ steps on average. Then it will continue by taking the same number of steps as the tame kangaroo. Therefore the total number of steps is

$$(1+\epsilon)\frac{N}{m} + 2(1+\epsilon)tm = (1+\epsilon)\sqrt{N}\left[\frac{1}{c} + 2ct\right]$$

To minimise the running time we minimise the term inside the square brackets to obtain $c = 1/\sqrt{2t}$. So adding up the number of tame and wild kangaroo steps gives the expected number of group operations in the worst

case

$$2(1+\epsilon)\sqrt{2tN}.$$

In the average case Q lies in the middle of the interval so the wild kangaroo only needs to bound across half of the interval so the running time is given by

$$(1+\epsilon)\sqrt{N}\left[\frac{2}{\sqrt{2t}}+\frac{\sqrt{t}}{\sqrt{2}}\right]=\frac{2+t}{\sqrt{2t}}(1+\epsilon)\sqrt{N}.$$

2. In the average case Q lies in the middle of the interval so the total number of steps is

$$(1+\epsilon)\frac{N}{2m}+2(1+\epsilon)tm=(1+\epsilon)\sqrt{N}\left[\frac{1}{2c}+2ct\right]$$

To minimise the running time we minimise the term inside the square brackets to obtain $c=1/(2\sqrt{t})$ and this leads to an average case expected running time of $2(1+\epsilon)\sqrt{tN}$ group operations and in the worst case an expected running time of $3(1+\epsilon)\sqrt{tN}$ group operations.

□

The memory requirement of this algorithm is very small as we only store the 'trap'. However the biggest issue with Pollard's kangaroo algorithm is that the running time depends on what probability of success we choose. For example with a probability of success of 0.63 the average case expected running time is a low as $2(1+\epsilon)\sqrt{N}$ but when the probability of success is 0.98 the average case expected running time is twice that, $4(1+\epsilon)\sqrt{N}$. Ideally we would like to have a probability of success close, if not equal to, 1 without compromising the overall

running time. In the next section we look at the method of van Oorschot and Wiener which has a probability of success close to 1, is easily parallelisable and has a very similar running time to that of Pollard's kangaroo algorithm when $t = 1$.

3.2 The van Oorschot & Wiener method

The van Oorschot and Wiener method [48] is a parallelised version of Pollard's kangaroo algorithm but through this process of parallelising the probability of solving the DLP in an interval increases close to 1.

3.2.1 Distinguished Points

Before we can describe the algorithm it is necessary to first understand what a distinguished point is. A distinguished point is an element of the group we are working in e.g. an elliptic curve point or an element of a finite field, which has a particular feature that is easily checked and confirmed. In practice these elements are stored as binary strings so for an element to be distinguished it may need to have a consecutive number of leading zero bits. Van Oorschot and Wiener, [48], and Benits, [2], take their distinguished points to have 32 leading zero bits followed by another 8 bits which have value less than or equal to 234. Therefore the probability that an element is a distinguished point is given by

$$\theta = \frac{234}{256} 2^{-32} = 234 \cdot 2^{-40}.$$

This seems quite arbitrary and we will see that the value of θ depends on the size of the space in which we are searching namely N.

3.2.2 Algorithm in detail

In Pollard's kangaroo algorithm we had two kangaroos one called Tame and the other Wild. Each of these walks were long as enough footprints needed to be made for the probability of solving the DLP in an interval to be significant. Instead if we have many short walks although the storage requirement will be greater we can easily parallelise the algorithm making it faster than Pollard's kangaroo algorithm. When a walk lands on a distinguished point they are stored on a server and is analogous to the 'trap' in Pollard's kangaroo algorithm. An obvious question at this point: Is the parallelised version of Pollard still low-memory? This answer to this is yes because we can set parameters is such a way that the storage requirement is constant, where that constant is actually the storage space that we have access to.

Let N_P be the number of processors. Half of the processors will work on tame walks (call these the Tame processors) and the other half will work on the wild walks (call these the Wild processors). Now we will describe the tame and wild walks that van Oorschot and Wiener used in more detail. Let S be the selection function as explained in Section 2.3.1 and U the set of step sizes of the pseudo-random walk. In the same way as Pollard's kangaroo algorithm we have a mean step size m which will be defined later.

A tame walk starts in the middle of the interval of possible exponents and each different tame walk starts a small distance, v, apart from each other but still

roughly in the middle of the interval of exponents. So the starting element of tame walk i is given by $x_1 = [iv]P_1$. We then apply the pseudorandom walk to x_1 and continue until we hit a distinguished point. At each step we increment the exponent and so the exponent at step k will be as follows

$$iv + \sum_{j=1}^{k-1} u_{S(x_j)}.$$

When the pseudorandom walk lands on a distinguished point the processor sends the element with its exponent to a server where it is stored in an easily accessible data structure i.e. a binary tree. The processor then continues the walk until it hits another distinguished point or is told to terminate the operation by the server.

A wild walk starts near Q and the different walks start a small distance, v apart i.e. the starting element of wild walk i is $y_1 = Q + [iv]P_1$. We then apply the pseudorandom walk function to y_1 and continue until we hit a distinguished point. At each step we increment the exponent and so the exponent before step k will be as follows

$$n_1 + iv + \sum_{j=1}^{k-1} u_{S(y_j)}.$$

When we land on a distinguished point, the element together with its multiple of P_1 is sent to the server for storage and again the processor continues its walk.

The server stores distinguished points discovered by the different walks in two separate, easily manageable data structures. Before the server stores a distinguished point it first checks whether that distinguished point is currently stored in the opposite data structure i.e. if a distinguished point is received from a tame

processor then the server checks whether that point is currently being stored in the data structure containing the distinguished points discovered from the wild walks and vice versa. We will assume searching and storing takes polynomial time and therefore will omit these operations from the Algorithm 4.

When we have a tame walk and a wild walk which meet/collide at the same point they follow the same path until they hit the same distinguished point. Let (R, a_t) and (R, b_w) be the distinguished points for a tame and wild processor respectively. Then we solve the DLP in an interval by simply calculating

$$n_1 = a_t - b_w \mod r.$$

3.2.3 Issues and Assumptions

Before we can look at the actual running time of the algorithm there are a number of issues that need to be dealt with. One of the key advantages to the van Oorschot and Wiener method is that it gives a linear speed up (by the number of processors) to Pollard kangaroo as well as giving a probability of success of close to 1. However as Algorithm 4 shows, the van Oorschot and Wiener method will continue until the DLP in an interval is solved so there is a possibility, although very small, that it will never terminate. In practice this does not occur.

Throughout the analysis of the van Oorschot and Wiener method as well as the Gaudry-Schost algorithm described in Section 4.1.2 we will need to know the expected number of steps in a walk. Recall that θ is the probability of an element or point being a distinguished point, therefore we have the following theorem:

Algorithm 4 The van Oorschot and Wiener method: Server side

INPUT: $P_1, Q \in G$ and number of processors N_P
OUTPUT: An integer n_1 such that $Q = [n_1]P_1$

Choose n_s to be the number of partitions of G
Choose randomly the function S as explained in Section 2.3.1
Choose randomly the jumps $U = \{u_0, \ldots, u_{n_s-1}\}$ and the spacing v
Initialise hash tables A_T and A_W to store distinguished points with their exponents from tame and wild walks respectively.
for $i \in \{1, \ldots, N_P/2\}$ **do**
    $start := iv$
    Initiate Tame processor i with $start$, n_s, S and U
    Initiate Wild processor i with $start$, n_s, S and U
end for
while DLP in an interval is not solved **do**
    if received tuple from Tame Processor i **then**
        Construct tuple as $z_T := (R, a_t)$
        if $(R, b_w) \in A_W$ for some b_w **then**
            Send terminate signal to all processors
            return $(a_t - b_w) \mod r$
        else if $z_T \in A_T$ **then**
            Send a random jump signal to the sender of z_T
        else
            Insert z_T into A_T
        end if
    else if received tuple from Wild Processor **then**
        Construct tuple as $z_W := (R, b_w)$
        if $(R, a_t) \in A_T$ for some a_t **then**
            Send terminate signal to all clients
            return $(a_t - b_w) \mod r$
        else if $z_W \in A_W$ **then**
            Send a random jump signal to the sender of z_W
        else
            Insert z_W into A_W
        end if
    end if
end while

Theorem 3.2.1. *Assuming that a pseudorandom walk behaves like a random walk then the expected number of steps in a pseudorandom walk before we land*

Algorithm 5 The van Oorschot and Wiener method: Tame Processor

INPUT: $P_1, Q \in G$, selection function S, jumps U and *start*

$a_1 := start$
$R := [a_1]P_1$
repeat
    $R := R + [u_{S(R)}]P_1$
    $a_{i+1} := a_i + u_{S(R)}$
    if R is a distinguished point **then**
        Send (R, a_t) to the server
        if Random jump signal received from server **then**
            $R := R + [u_{\text{Random}}]P_1$
        end if
    end if
until Terminate signal from server

Algorithm 6 The van Oorschot and Wiener method: Wild Processor

INPUT: $P_1, Q \in G$, $N_1, n_s \in \mathbb{N}$, selection function S, jumps U and *start*

$b_1 := start$
$R := Q + [b_1]P_1$
repeat
    $R := R + [u_{S(R)}]P_1$
    $b_{i+1} := b_i + u_{S(R)}$
    if R is a distinguished point **then**
        Send (R, b_w) to the server
        if Random jump signal received from server **then**
            $R := R + [u_{\text{Random}}]P_1$
        end if
    end if
until Terminate signal from server

on a distinguished point is $1/\theta$.

Proof. Let Y be the random variable for the number of steps before we expect to hit a distinguished point. Then the expectation of Y is

$$\mathbb{E}(Y) = \sum_{i=1}^{\infty} i(1-\theta)^{i-1}\theta = \theta \sum_{i=1}^{\infty} i(1-\theta)^{i-1}$$

If we let $S = \sum_{i=1}^{\infty} i(1-\theta)^{i-1}$ then we can simplify S as follows

$$\begin{aligned} S - (1-\theta)S &= \sum_{i=0}^{\infty}(i+1)(1-\theta)^i - \sum_{i=1}^{\infty} i(1-\theta)^i \\ &= 1 + \sum_{i=1}^{\infty}(1-\theta)^i \\ &= 1 + \frac{1-\theta}{\theta} \\ &= \frac{1}{\theta}. \end{aligned}$$

Therefore $S = \frac{1}{\theta^2}$ and so we can now find the expectation of Y which is given by

$$\mathbb{E}(Y) = \theta S = \frac{1}{\theta}.$$

□

There is a very small possibility that a walk never reaches a distinguished point so to combat this van Oorschot and Wiener defined a maximum number of steps of a walk before it is abandoned. They chose the maximum number of steps to be $20/\theta$. Although this seems arbitrary, it stops processing time being wasted in a never ending walk. In the analysis of their algorithm this is not very important but we will show in Section 4.1.6 (Lemma 4.1.9) that the proportion of walks which never reach a distinguished point using the limit of $20/\theta$ steps is bounded above by 5×10^{-8}. This is a very small proportion and again in practice this does not cause any problems.

Another possibility when having numerous walks of one type is that two walks of one type collide and continue to the same distinguished point.

Definition 3.2.2. The different collisions between the walks will be defined as follows: -

- A **TT collision** is a collision resulting from two Tame walks terminating at the same distinguished point,
- A **WW collision** is a collision resulting from two Wild walks terminating at the same distinguished point,
- A **TW collision** is a collision resulting from a Tame and Wild walk terminating at the same distinguished point.

TT and WW collisions are a problem as we are starting walks of the same type quite close to each other therefore there is a probability that they will collide. Teske [45] has shown that the expected number of collisions within the same type of walks is 2. Therefore in practice this problem can be ignored. We summarise these points in Heuristic 3.2.3.

Heuristic 3.2.3. Suppose N_1 is sufficiently large, the selection function is chosen at random and θ is chosen appropriately then it is expected that a walk will reach a distinguished point in $1/\theta$ steps and footprints of walks of the same type are mutually independent.

When we look at the Gaudry-Schost algorithm TT and WW collisions will be dealt with more rigourously.

3.2.4 Running Time Analysis

In a similar fashion to Pollard's kangaroo algorithm we optimise the running time dependent on whether we expect to see more worst case or average case problem instances. We will use Pseudorandom walk 2 to analyse the running time of the van Oorschot and Wiener method and give the relevant constants in this setting.

Theorem 3.2.4. *Using the method of van Oorschot and Wiener and Pseudorandom walk 2 to solve the DLP in an interval of size N and assuming Heuristic 3.2.3 holds,*

1. *If we optimise for the worst case then $c_1 = N_P/\sqrt{8}$ and $m = N_P\sqrt{N/8}$. Therefore the average case expected running time is $2.12(1+\epsilon)\sqrt{N}/N_P + 1/\theta$ group operations and the worst case expected running time is $2.83(1+\epsilon)\sqrt{2N}/N_P + 1/\theta$ group operations.*

2. *If we optimise for the average case then $c_1 = N_P/4$ and $m = N_P\sqrt{N}/4$. Therefore the average case running time is $2(1+\epsilon)\sqrt{N}/N_P + 1/\theta$ group operations and the worst case running time is $3(1+\epsilon)\sqrt{N}/N_P + 1/\theta$ group operations.*

N.B. All of these running times are 'per processor'.

Proof. In the van Oorschot and Wiener method we do not know where the wild walks are in relation to the tame walks. They could be in front of or, equally, behind the tame walks. Therefore we consider the walks of one type as a 'herd' so we now have a front and back herd.

1. In the worst case the distance (in exponents) between the front and back herd is $N/2$. It is expected to take $N/2m$ steps for the back herd to reach the starting point of the front herd. Let us now consider the region already visited by the front herd and where the back herd are now stepping. In every interval of length m we expect $N_P/2$ footprints of the front herd (assuming each walk is independent as given by Heuristic 3.2.3). The probability that one of the back herd lands in a footprint of the front herd is then $N_P/2m$. Again assuming Heuristic 3.2.3 holds, the probability that one of the walks of the back herd lands in one of these footprints is

$$\frac{N_P}{2m} \cdot \frac{N_P}{2} = \frac{N_P^2}{4m}.$$

Therefore the expected number of steps before a collision occurs is $4m/N_P^2$. A collision is only detected once a distinguished point is visited so we expect each processor to make a further $1/\theta$ steps. Putting this all together, in the worst case we expect each processor to make the following number of group operations before a collision

$$\frac{N}{2m} + (1+\epsilon)\frac{4m}{N_P^2} + \frac{1}{\theta}. \tag{3.2}$$

Taking $\epsilon = 0$ we can minimise equation (3.2) by taking $c_1 = N_P/\sqrt{8}$ and therefore the mean step size $m = N_P\sqrt{N/8}$ so in the worst case the expected running time is $2.83(1+\epsilon)\sqrt{N}/N_P + 1/\theta$ group operations per processor. In the average case the distance between the front herd and the back herd is $N/4m$ so the expected running time is $2.12(1+\epsilon)\sqrt{N}/N_P + 1/\theta$ group operations per processor.

2. In the average case the distance between the front and back herds is $N/4m$. Assuming Heuristic 3.2.3 holds, we expect the following number of group operations before a collision

$$\frac{N}{4m} + (1+\epsilon)\frac{4m}{N_P^2} + \frac{1}{\theta}. \tag{3.3}$$

Taking $\epsilon = 0$ we can minimise equation (3.3) by taking $c_1 = N_P/4$ and therefore the mean step size $m = N_P\sqrt{N}/4$ so in the average case the expected running time is $2(1+\epsilon)\sqrt{N}/N_P + 1/\theta$ group operations per processor. Using this mean step size in the worst case gives an expected running time of $3(1+\epsilon)\sqrt{N}/N_P + 1/\theta$ group operations per processor.

□

We expect to store N_P distinguished points when using the van Oorschot Wiener method to solve the DLP in an interval which is still constant storage. In addition this method gives a linear speedup (by the number of processors) to the kangaroo algorithm by distributing the work between clients but due to its very nature the number of clients needs to be known before the method starts. As a result one is not able to take advantage of more processors if they become available. We will next see an algorithm by Gaudry and Schost [19] which does not require the knowledge of the number of clients prior to starting and in addition can easily utilise extra clients if they become available.

3.3 The Tame-Wild Birthday Paradox

In this section we will revise a slight alteration to the Birthday Paradox which is used by Gaudry and Schost [19] to analyse their algorithm.

Menezes, van Oorschot and Vanstone [30, Fact 2.28] describe this problem as follows: Suppose that there are two urns, one containing M white balls numbered 1 to M, and the other containing M red balls numbered 1 to M. First, m_1 balls are selected from the first urn and their numbers listed. Then m_2 balls are selected from the second urn and their numbers listed. Finally, the number of coincidences between the two lists are counted. We consider the setting where the balls from both urns are drawn one at a time, with replacement.

Theorem 3.3.1. *Given that we are picking the balls uniformly at random, the expected number of selections that need to be made in total before we have a coincidence between the lists is*

$$\sqrt{\pi M} + O(1).$$

Sketch of proof. Nishimura and Sibuya [31] prove that the probability of zero coincidences after m_1 steps in urn 1 and m_2 steps in urn 2 is given by,

$$\mathbb{P}(M, m_1, m_2) = \frac{1}{M^{m_1+m_2}} \sum_{t_1=0}^{m_1} \sum_{t_2=0}^{m_2} M^{(t_1+t_2)} \left\{ \begin{array}{c} m_1 \\ t_1 \end{array} \right\} \left\{ \begin{array}{c} m_2 \\ t_2 \end{array} \right\}. \tag{3.4}$$

If we let the number of selections be equal and of the following order $m_1 = m_2 = m = O(\sqrt{M})$ and $M \to \infty$ then Nishimura and Sibuya showed that the above

probability tends to the following,

$$\mathbb{P}(M,m,m) \to \exp\left(-\frac{m^2}{M}\left[1+O\left(\frac{1}{\sqrt{M}}\right)\right]\right) \approx \exp\left(-\frac{m^2}{M}\right). \tag{3.5}$$

Let X be the random variable of the number of selections from urn 1 before we have a coincidence between the lists. Then the probability in equation (3.5) can be denoted by $\mathbb{P}(X > m)$. We can use this probability to calculate the expectation of X which is

$$\begin{aligned}
\mathbb{E}(X) &= \sum_{l=1}^{\infty} l\mathbb{P}(X=l) \\
&= \sum_{l=1}^{\infty} l\mathbb{P}(X>l-1) - \sum_{l=1}^{\infty} l\mathbb{P}(X>l) \\
&= \sum_{l=0}^{\infty} (l+1)\mathbb{P}(X>l) - \sum_{l=1}^{\infty} l\mathbb{P}(X>l) \\
&= \sum_{l=0}^{\infty} \mathbb{P}(X>l) \\
&\approx \sum_{l=0}^{\infty} \exp\left(-\frac{l^2}{M}\right).
\end{aligned}$$

As $\exp\left(-\frac{l^2}{M}\right)$ is monotonically decreasing and takes values in $[0,1]$, the maximum difference between this sum and its corresponding integral is at most 1. Therefore we can can instead estimate this sum as

$$\int_{x=0}^{\infty} \exp\left(-\frac{x^2}{M}\right)dx = \sqrt{M}\int_{u=0}^{\infty} \exp\left(-u^2\right)du = \frac{\sqrt{\pi M}}{2}. \tag{3.6}$$

This equation gives us an approximation of the number of selections from urn 1 or 2 before we can expect a coincidence between the lists. Therefore the total

number of selections is twice this which completes the proof. □

Henceforth we will use the term Tame-Wild Birthday Paradox to denote this problem with its associated running time.

Chapter 4

New Methods for Computing Discrete Logarithms in Intervals

In this chapter we look at the new low-memory algorithms for solving the DLP in an interval. These algorithms utilise the Tame-Wild Birthday Paradox(as described in Section 3.3) and are explained in their order of running times which reduce as we progress.

- In Section 4.1 we start off with an explanation of the standard algorithm of Gaudry and Schost which is not as fast as Pollard's kangaroo method.
- We make a small improvement to the Gaudry-Schost algorithm in Section 4.1.5 which will become useful in Chapter 6.
- Then in Section 4.2 we present Pollard's very recent (and as a result of a personal communication with us) improvement to his kangaroo algorithm which does not currently appear anywhere in the literature. These come in

the three and four kangaroo variants.

- Finally we put together the new ideas of Pollard and our previous improvement of the Gaudry-Schost algorithm in Section 4.3 to have three new variants of the Gaudry-Schost algorithm namely the 3 set, 4 set and improved 4 set variants.

Together with Pollard we are in the process of writing a research paper on his improved kangaroo algorithm and our improved Gaudry-Schost algorithm [13]. These two algorithms are now the fastest algorithms for solving the DLP in an interval.

4.1 The Gaudry-Schost algorithm

Gaudry and Schost [19] developed an algorithm to solve the DLP in an interval using a different type of analysis to that of the Pollard kangaroo algorithm [34, 35]. More precisely they use the Tame-Wild Birthday Paradox to analyse the expected running time of their algorithm. The Gaudry-Schost algorithm is designed principally to be a distributed across a number of processors in a similar way to the method of van Oorschot and Wiener [46, 48].

4.1.1 Tame and Wild Sets

We have defined tame and wild walks in Section 3.2 but for the Gaudry-Schost algorithm we need to specify some more about the sets in which they walk. From Definition 3.0.9 we know that

$$Q \in \{[a_1]P_1 \mid a_1 \in [-N_1, N_1]\}.$$

Rather than looking at the set of possible group elements we will consider these sets in terms of their exponents.

Definition 4.1.1. Given the DLP in an interval as in Definition 3.0.9 then define

$$T = \{a_1 \mid a_1 \in [-N_1, N_1]\}.$$

We call this the 'Tame' set.

The exponent of Q lies somewhere in the tame set, T. We then centre a similar set of the same cardinality around n_1 where $Q = [n_1]P_1$.

Definition 4.1.2. Given the DLP in an interval as in Definition 3.0.9 then define

$$W = n_1 + T$$

We call this the 'Wild' set.

In the case of the wild set, an element $w \in W$ refers to the group element $Q + [w]P_1$.

4.1.2 The algorithm in brief

The Gaudry-Schost algorithm proceeds as follows. Just as in the van Oorschot and Wiener algorithm we have N_P processors. Half of the processors will work on tame walks (call these the Tame processors) and the other half will work on the wild walks (call these the Wild processors). Each tame or wild processor chooses a random starting point in T or W respectively and performs a pseudorandom walk until a distinguished point is hit. This distinguished point is then stored together with its exponent on a server and is analogous to the 'trap' which is set in Pollard's kangaroo algorithm [11, 35]. The processor then chooses another random starting point and repeats the task. After a certain number of steps (elements/points visited) have been taken we expect to find a collision between a tame processor and a wild one. If this occurs on a point that is not distinguished then the two walks will continue on the same path to the same distinguished point. As this distinguished point is stored after the first walk then seen again on the second walk we find a collision and the server can solve for the discrete logarithm of Q. For the purposes of presenting the algorithm we assume there is a function

$\mathsf{walk}(x_i, a_i)$ which computes the next step in the random walk and returns the tuple (x_{i+1}, a_{i+1}). Also we will store the distinguished points in an easily searched storage structure such as a hash table. So we will assume searching and storing times are constant and therefore will omit these operations from Algorithm 7. In Algorithms 8 and 9 we refer to a maximum walk length which is discussed in Section 4.1.6.

Algorithm 7 The Gaudry-Schost Algorithm: Server side

INPUT: $N_1 \in \mathbb{N}$, $P_1, Q \in G$
OUTPUT: Integer tuple n_1 such that $Q = [n_1]P_1$

```
 1: Choose the function walk uniformly at random
 2: Let A_T be a hash table to store distinguished points from tame walks
 3: Let A_W be a hash table to store distinguished points from wild walks
 4: while DLP in an interval not solved do
 5:     if received tuple from Tame Processor then
 6:         Construct tuple as z_T := (R, a_t)
 7:         if (R, a_w) ∈ A_W for some a_w then
 8:             Send terminate signal to all clients
 9:             return (a_t − a_w) mod r
10:         else
11:             Insert z_T into A_T
12:         end if
13:     else
14:         Construct tuple as z_W := (R, a_w)
15:         if (R, a_t) ∈ A_T for some a_t then
16:             Send terminate signal to all clients
17:             return (a_t − a_w) mod r
18:         else
19:             Insert z_W with A_W
20:         end if
21:     end if
22: end while
```

Algorithm 7, 8 and 9 presents a brief outline of the Gaudry-Schost algorithm when applied to the DLP in an interval. However a number of issues need to be resolved.

Algorithm 8 The Gaudry-Schost Algorithm: Tame Processor

INPUT: $N_1 \in \mathbb{N}$, $P_1, Q \in G$, function walk

repeat
  Choose a random integer $a_1 \in [-N_1, N_1]$
  $x_1 := [a_1]P_1$, Counter:= 0
  while x_i is not a distinguished point and Counter $\leq 20/\theta$ **do**
    Counter++
    $(x_{i+1}, a_{i+1}) := \mathsf{walk}(x_i, a_i)$
  end while
  Send (x_t, a_t) to the server.
until The server sends terminate signal

Algorithm 9 The Gaudry-Schost Algorithm: Wild Processor

INPUT: $N_1 \in \mathbb{N}$, $P_1, Q \in G$, function walk

repeat
  Choose a random integer $a_1 \in [-N_1, N_1]$
  $y_1 := Q + [a_1]P_1$, Counter:= 0
  while y_i is not a distinguished point and Counter $\leq 20/\theta$ **do**
    Counter++
    $(y_{i+1}, a_{i+1}) := \mathsf{walk}(y_i, a_i)$
  end while
  Send (y_w, a_w) to the server.
until The server sends terminate signal

1. What will our deterministic pseudorandom walk look like?

2. What is the expected number of steps before we see a collision?

3. How do we decide the value of θ?

4. How many steps will be needed in practice i.e. are there any wasted or 'bad' steps?

The average expected running time of the Gaudry-Schost algorithm for solving the DLP in an interval of size N is approximately $2.08\sqrt{N}$ group operations. Gaudry and Schost [19, Section 3.1] give a proof of this but they do not give any discussion

on 'bad' walks or steps. There are two types of bad steps that we have to consider. The first type comes from walks which never reach a distinguished point (type 1) and the second type comes from walks which step outside of the tame and wild sets (type 2). Our analysis will be able to deal with walks anywhere in T or W but walks outside of these sets cannot be dealt with. So if we allow walks to start near the edges of the tame or wild sets then these could be wasted. To resolve this we do not start walks near the edges of the tame and wild sets and the subsets of T and W where walks cannot start will be discussed in Section 4.1.6. First we will analyse the original Gaudry-Schost algorithm and then investigate whether there are any improvements possible which make this algorithm competitive with Pollard's kangaroo algorithm or the method of van Oorschot and Wiener.

4.1.3 The Original Gaudry-Schost algorithm

We now apply the original Gaudry-Schost algorithm to solve the DLP in an interval of size N. We will continue to use elliptic curve notation, for clarity, let $G = E(\mathbb{F}_q)[r]$ and so let $P_1, Q \in G$ where P_1 is a generator of the group. We want to find an integer n_1 such that $Q = [n_1]P_1$ where we have some extra information that $n_1 \in [-N_1, N_1]$ where N_1 is less than the order of G. Let $N = 2N_1 + 1$.

Let $\mathcal{A} = T \cap W$ be the set of exponents that are common to both the tame and wild sets and let M be the cardinality of $\mathcal{A}$. We will use the Tame-Wild Birthday Paradox (Section 3.3) to solve for the expected number of steps before we have a TW collision. However a TW collision can ONLY occur in the overlap of T and W namely $\mathcal{A}$. The tame steps are analogous to the selections from urn 1 and the wild steps are analogous to the selections from urn 2. However the problem with

using Theorem 3.3.1 to analyse the Gaudry-Schost algorithm is that we will not be picking the elements in T and W uniformly at random. Although the starting point of the pseudorandom walks will be random, inherently the rest of the steps will not be random so we only obtain a heuristic running time. Later we give experimental results which show that the pseudorandom walks get close enough to random selection. The pseudorandom walk we will use in the Gaudry-Schost algorithm to solve the DLP in an interval is described in Pseudorandom walk 3.

Pseudorandom Walk - 1D 3 (Gaudry-Schost)**.** The pseudorandom walk proceeds as follows,

1. We have a selection function S as described in Section 2.3.1 which takes as input the current footprint of the pseudorandom walk and outputs the index of the partition of G which that element belongs to.

2. We define a mean step size m such that $m/\theta = \sqrt{N}$ so that the subset of the interval where walks are not permitted to start is only of size approximately $\sqrt{N}$ as we will see in Section 4.1.6.

3. For $0 \leq j < n_s$ we choose random step sizes $1 \leq u_j \leq 2m$ and then precompute $[u_j]P_1$.

4. The pseudorandom walk is as follows

$$x_{i+1} = f(x_i) = x_i + [u_{S(x_i)}]P_1.$$

   We update the exponent as follows $a_{i+1} = a_i + u_{S(x_i)}$.

The expected length of a walk i.e. the distance traveled by a walk in terms of exponents is given by Lemma 4.1.3

Lemma 4.1.3. *The expected length of a pseudorandom walk before we land on a distinguished point is* $\frac{m}{\theta}$.

Proof. Let Y be the random variable for the number of steps before we expect to hit a distinguished point. From Theorem 3.2.1 we know that $E(Y) = \frac{1}{\theta}$. Now if we let Z be the random variable of the size of each step in terms of the exponents then $E(Z) = m$. The random variables Y and Z are independent so the expected length of a walk is the expectation of the product of Y and Z so

$$E(YZ) = E(Y) \cdot E(Z) = \frac{1}{\theta} \cdot m = \frac{m}{\theta}.$$

□

Unlike the pseudorandom walk that is used in Pollard's kangaroo algorithm we do not want our walks to travel a long distance. The reason for this is that in our analysis we do not want walks to jump outside of the tame and wild sets. In addition we do not want our walks to be too localised otherwise Heuristic 4.1.4 will not hold hence part 2 of Pseudorandom walk 3. The interval where walks are not permitted to start will be the distance given in Lemma 4.1.3, so in order to keep this small we define m as shown in part 2 of Pseudorandom walk 3.

To counteract walks jumping outside of T and W we will not start walks within m/θ of the right hand edge of T and W. Therefore our tame walks start with

$x_1 = [a_1]P$ for some $a_1 \in [-N_1, N_1 - m/\theta]$ and continue as follows

$$x_{i+1} = f(x_i) = x_i + [u_{S(x_i)}]P.$$

The wild walks start with $y_1 = Q + [a_1]P$ for some $a_1 \in [-N_1, N_1 - m/\theta]$ and continues as follows

$$y_{i+1} = f(y_i) = y_i + [u_{S(y_i)}]P.$$

If $a_1 = N_1 - m/\theta$ for any walk then there is a 50% chance that walk will jump outside of T or W. This probability drops as a_1 reduces so we can limit the expected number of steps wasted in this way. In Section 4.1.6 we look at this in more depth. Due to not starting walks within m/θ of the right hand edge of T and W we have a further problem of selecting elements uniformly in $(N_1 - m/\theta, N_1]$. In practice as the set where walks are not allowed to start is small in comparison to the size of the interval this does not cause any problems.

4.1.4 Running time analysis

Both Pollard's kangaroo algorithm and the method by van Oorschot & Wiener use the mean step size to optimise the running times of their algorithms. In the Gaudry-Schost approach we use a Tame-Wild Birthday Paradox analysis on the overlap of the tame and wild sets, $\mathcal{A}$, to obtain an expected running time. Therefore the mean step size is not important to the running time analysis, however as shown above and in Section 4.1.6 it will be constrained when we consider the wasted or 'bad' steps, which cannot be included in the analysis. If N is sufficiently large then we can assume Heuristic 2.3.3 and we have the

following,

Heuristic 4.1.4. If N is sufficiently large then Pseudorandom walk 3 is sufficiently random that the Tame-Wild Birthday Paradox analysis applies to the set $\mathcal{A}$.

Similarly to both Pollard's kangaroo algorithm and the method of van Oorschot & Wiener, the running time of the Gaudry-Schost algorithm depends on the location of Q within the tame set, T. To compare between the different algorithms for solving the DLP in an interval of size N it is easier if we take the number of processors to be 1 i.e. $N_P = 1$. Now we look at the two extreme cases.

4.1.4.1 The best case

Lemma 4.1.5. *Assuming Heuristic 4.1.4, the expected number of group operations to solve the DLP in an interval of size N using the original Gaudry-Schost algorithm in the best case (without considering the bad steps) is*

$$\sqrt{\pi N} + \frac{1}{\theta}. \tag{4.1}$$

Proof. In the best case Q lies in the middle of the interval i.e. $Q = [0]P$, then $\mathcal{A} = T = W$ and $M = N$. So every step taken by a tame or wild walk is in the overlap $\mathcal{A}$. Therefore using the Tame-Wild Birthday Paradox analysis from Theorem 3.3.1, the number of steps before we expect to see a collision is $\sqrt{\pi N}$. Then the final walk will need to continue to the distinguished point already being stored and we have the result. □

4.1.4.2 The worst case

Lemma 4.1.6. *Assuming Heuristic 4.1.4, the expected number of group operations to solve the DLP in an interval of size N using the original Gaudry-Schost algorithm in the worst case (without considering the bad steps) is*

$$\sqrt{2\pi N} + \frac{1}{\theta}. \tag{4.2}$$

Proof. In the worst case Q lies at the end of the interval (e.g., $Q = [-N_1]P$), then $M = |T|/2 = |W|/2 = N/2$. So on average only half of each of the tame and wild walks are in $\mathcal{A}$. Therefore using the Tame-Wild Birthday Paradox analysis from Theorem 3.3.1, the number of steps in the overlap before we expect to see a collision is $\sqrt{\pi N/2}$. So the expected number of group operations before we have a TW collision is given by

$$2\sqrt{\pi N/2} = \sqrt{2\pi N}.$$

Then the walk will need to continue to the distinguished point already being stored and we have the result. □

4.1.4.3 The average case

Theorem 4.1.7. *Assuming Heuristic 4.1.4, the average expected number of group operations to solve the DLP in an interval of size N using the original Gaudry-Schost algorithm (without considering the bad steps) is approximately*

$$2.08\sqrt{N} + \frac{1}{\theta}. \tag{4.3}$$

Proof. To obtain an average case expected running time we need to average out over all possibilities of Q. Let us consider $Q = [-N_1 + xN]P$ where $x \in [0, 1/2]$ then the overlap $\mathcal{A}$ is of size $M = (1/2 + x)N$. The proportion of steps or walks in the overlap is $(1/2 + x)$ therefore the we expect to make $(1/2 + x)^{-1}$ selections from the tame and wild sets to have 1 step in the overlap. So we have

$$\begin{aligned} & 2\int_{x=0}^{1/2} \frac{1}{1/2+x}\sqrt{\pi(1/2+x)N}\, dx \\ &= 2\sqrt{\pi N}\int_{x=0}^{1/2} \frac{1}{(1/2+x)^{1/2}}\, dx \\ &= 4\sqrt{\pi N}\left(1-(1/2)^{1/2}\right) \\ &\approx 2.076559\sqrt{N}. \end{aligned}$$

Then the walk will need to continue to the distinguished point already being stored and we have the result. □

As the running time of the different algorithms depends of the position of Q within the tame set, T, it would be greatly beneficial if we could somehow have a constant running time regardless of the position of Q. We deal with this situation in the next section.

4.1.5 Improving the Gaudry-Schost Algorithm

The running time of the original Gaudry-Schost algorithm depends on the size of the overlap between the tame and wild sets. If it is possible to make the size of this overlap constant for all possible Q then the expected running time will be constant for all problem instances. One way to approach this is to make the

worst case (where Q lies at the end of the interval $[-N_1, N_1]$) better. We start our random walks uniformly in some subset of the search space of size kN, rather than over the whole space, where $0 < k < 1$. We call this new approach the 'Improved Gaudry-Schost algorithm'. Note that when $k = 1$ we have the original Gaudry-Schost approach. In the tame set we will search through kN elements centred around the middle of the interval. Therefore the 'New Tame' set is,

$$T' = \{a_1 \mid a_1 \in [-kN_1, kN_1]\}.$$

In the wild set we will split the new search space into two equal parts at either ends of the interval so the 'New Wild' set is,

$$W' = \{n_1 + a_1 \mid a_1 \in [-N_1, -N_1(1-k)] \cup [(1-k)N_1, N_1]\}.$$

W' is the union of two disjoint sets which are given below,

$$W'_1 = \{n_1 + a_1 \mid a_1 \in [-N_1, (k-1)N_1]\}$$
$$W'_2 = \{n_1 + a_1 \mid a_1 \in [(1-k)N_1, N_1]\}.$$

Let $\mathcal{A}' = T' \cap W'$ denote the overlap between the new tame and wild sets and M' its cardinality. In the previous best case Q lies in the middle of the interval so the cardinality of $\mathcal{A}'$ is $M' = (2k-1)N$. In the previous worst case Q lies at the

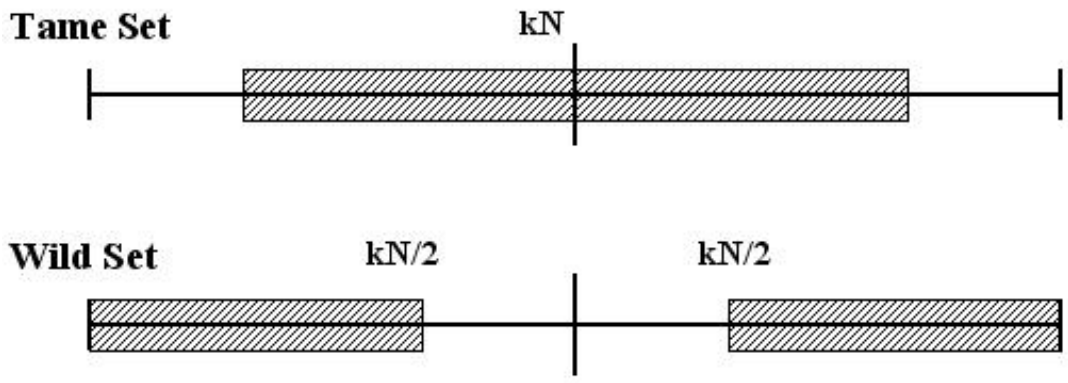

Figure 4.1: Searching kN of the Tame and Wild Sets

end of the interval so $M' = \frac{kN}{2}$. Equating these two different values of M' we get

$$\begin{aligned} 2k - 1 &= \frac{k}{2} \\ k &= \frac{2}{3}. \end{aligned}$$

Theorem 4.1.8. *Assuming Heuristic 4.1.4, the expected number of group operations to solve the DLP in an interval of size N (without considering the bad steps) using the improved Gaudry-Schost algorithm with $k = \frac{2}{3}$ is*

$$\sqrt{\frac{4}{3}\pi N} + \frac{1}{\theta} \approx 2.05\sqrt{N} + \frac{1}{\theta}. \tag{4.4}$$

Proof. When $k = 2/3$ we can see in Figure 4.1 that for all problem instances (different positions of Q) the overlap has cardinality

$$|\mathcal{A}'| = M' = \frac{N}{3}.$$

On average half of all walks are in the overlap $\mathcal{A}'$ just as in the previous worst case.

However the new tame and wild sets are smaller than in the original algorithm so the expected number of steps before we have a TW collision is given by

$$2\sqrt{\pi\frac{N}{3}} = \sqrt{\frac{4}{3}\pi N}.$$

Then the walk will need to continue to the distinguished point already being stored and we have the result. □

This is an improvement on the original Gaudry-Schost algorithm for solving the DLP in an interval (Theorem 4.1.7) with the added bonus that the worst and average cases are the same. In the next section we will look at the wasted or 'bad' steps which cannot be counted in the Tame-Wild Birthday Paradox analysis and therefore must be added to the expected running time.

4.1.6 Counting bad steps in the Gaudry-Schost Algorithm

In this section we give bounds on the number of wasted or 'bad' steps. The first type comes from a walk which never reaches a distinguished point and the second type comes from walks which step outside of the tame and wild sets and therefore cannot be added to the Tame-Wild Birthday Paradox.

4.1.6.1 Bad steps of type 1

To guard against walks which get stuck in a loop i.e. hitting an element that has already been visited in that walk, or walks which never reach a distinguished point van Oorschot & Wiener [48] set a maximum number of steps in a walk. They take

the maximum number of steps to be 20 times as long as the expected number of steps in a walk. We know from Theorem 3.2.1 that the expected number of steps is $\frac{1}{\theta}$. Let $\mathcal{T}$ denote the maximum number of steps in a walk which we will also set to be $\frac{20}{\theta}$. Van Oorschot and Wiener gave the following lemma and proof.

Lemma 4.1.9. *If we take $\mathcal{T} = \frac{20}{\theta}$ then the proportion of steps which cannot be included in the Tame-Wild Birthday Paradox analysis due to walks which never reach a distinguished point is bounded by 5×10^{-8}.*

Proof. The probability that a walk never reaches a distinguished point is bounded as follows

$$(1-\theta)^{20/\theta} \leq (\exp(-\theta))^{20/\theta} = \exp(-20).$$

Each abandoned walk is 20 times longer than the average walk so the proportion of bad steps of type 1 is approximately $20\exp(-20) < 5 \times 10^{-8}$. □

This shows that the proportion of bad steps of type 1 are very small and only has a minute effect on the expected running time.

4.1.6.2 Bad steps of type 2

As the new tame set, T', and wild set, W', are not of the same form the bad steps of type 2 will be calculated differently for the tame and wild walks. Nevertheless the number of bad steps of type 2 from the tame walks is less than half the number from wild walks as there are two disjoint sets that the pseudorandom walk start in.

Definition 4.1.10. Let B denote the number of bad steps of type 2.

We now need to define the maximum step size. From Pseudorandom walk 3 we know that the largest step is no bigger than $2m$ so we will take this to be our maximum step size.

When solving the DLP in an interval the pseudorandom walks move in a positive direction (to the right) in terms of the exponents so starting a walk near the righthand edge of the sets T', W_1' or W_2' will mean that these walks are highly likely to reach a distinguished point outside of the sets. Whilst in practice these distinguished points can be used to get a TW collision we cannot use walks that step outside of the sets in our Tame-Wild Birthday Paradox analysis. This is because we can only apply a Tame-Wild Birthday Paradox analysis to a set of known size i.e. $|\mathcal{A}| = M$ so steps outside of this set cannot be counted even though in practice any such distinguished point will be stored on the server. Therefore by Lemma 4.1.3 we will not start a walk on an element with exponent less than $\frac{m}{\theta}$ from the righthand edge of T', W_1' or W_2'.

Lemma 4.1.11. *Given that the algorithm has made K steps, then we can bound the number of bad steps of type 2 as,*

$$B < \frac{39m}{2N\theta/3 - 2m} \cdot 30K = \frac{585K}{N\theta/3 - m} \tag{4.5}$$

Proof. Let X denote the set of all elements of T' such that if a pseudorandom walk starts at those elements there is a nonzero probability that the walk will step outside of the search space. Then

$$X = \left\{ a_1 \;\middle|\; a_1 \in \left[\frac{2N_1}{3} - \frac{20 \cdot 2m}{\theta}, \frac{2N_1}{3} - \frac{m}{\theta} \right] \right\}.$$

A crude bound on the number of bad steps of type 2 that occur from tame walks can be given by the following calculation: the probability that a tame walk starts in X $\times$ the expected number of tame walks $\times\mathcal{T}$. So the probability that a walk starts in X is

$$\frac{\frac{40m}{\theta}-\frac{m}{\theta}}{\frac{2N}{3}-\frac{m}{\theta}}=\frac{39m}{2N\theta/3-m}.$$

Given that the algorithm has made K steps, the expected number of tame walks is just the total number of tame steps, $K/2$, multiplied by the probability of a point being distinguished namely θ. Therefore we have a bound for the tame walk component of B which is

$$\frac{39m}{2N\theta/3-m}\times\frac{K\theta}{2}\times\frac{20}{\theta}=\frac{39m}{2N\theta/3-m}\cdot 10K.$$

For the wild walks we do the same calculations for W_1' and W_2'. A bound for each of these sets will be the same so let us consider W_1' only. Let Y denote the set of all elements of W_1' such that if a pseudorandom walk starts at those elements there is a nonzero probability that the walk will step outside of the search space. Then

$$Y=\left\{a_1 \;\middle|\; a_1\in\left[-\frac{N_1}{3}-\frac{40m}{\theta},-\frac{N_1}{3}-\frac{m}{\theta}\right]\right\}.$$

The probability of a walk starting in Y is

$$\frac{\frac{40m}{\theta}-\frac{m}{\theta}}{\frac{N}{3}-\frac{m}{\theta}}=\frac{39m}{N\theta/3-m}.$$

So a bound for the W_1' component of B is

$$\frac{39m}{N\theta/3-m}\times\frac{K\theta}{4}\times\frac{20}{\theta}=\frac{39m}{2N\theta/3-2m}\cdot 10K.$$

This is marginally larger than the tame walk component of B so multiplying this by 3 takes into account both the tame and wild walk components of B hence proving the result. □

We can see from Lemma 4.1.11 that the bound on B depends upon N, θ and m. Another way to look at it is: if we want to bound B as a certain proportion of the total number of steps (i.e., 1%) then for a specific N there may be limits to how large θ and m can be. Table 4.1 gives the minimum bounds for N if we want to keep B as 1% of the total number of steps for different values of θ and m. For smaller N we cannot have too small a probability of an element being distinguished or too large a mean step size.

	$\theta = 2^{-8}$	$\theta = 2^{-16}$	$\theta = 2^{-32}$
$m = 32$	2^{32}	2^{40}	2^{56}
$m = 4096$	2^{40}	2^{48}	2^{64}
$m = 52428$	2^{44}	2^{52}	2^{68}
$m = 134217728$	2^{56}	2^{64}	2^{80}

Table 4.1: Minimum values of N to have $B < 1\%$ of the total number of steps for different θ and m

When deciding on the value of θ another consideration needs to be made. If we let K be the total number of steps of the algorithm before we can solve for the DLP in an interval then we expect to have made $K\theta$ walks. Therefore we expect to store $K\theta$ distinguished points. In order to use only constant storage as $K \approx 2\sqrt{N}$ for all of the Gaudry-Schost and kangaroo algorithms we must take $\theta = c/\sqrt{N}$ where c is approximately half of the storage space available.

4.1.7 The complete improved running time

To conclude the Gaudry-Schost algorithm in this setting we put the bad steps together with the Tame-Wild Birthday Paradox analysis to get an expected running time.

Lemma 4.1.12. *The expected number of group operations before we can solve for the DLP in an interval of size N using the improved Gaudry-Schost algorithm and bounding the proportion of bad steps to 1% of the total steps, is*

$$2.07\sqrt{N} + \frac{1}{\theta}.$$

Proof. Let K be the total number of steps. Then K must include enough steps to get a TW collision, to counteract those walks which never reach a distinguished point and those walks which step outside the tame and wild sets. For clarity let us exclude the $\frac{1}{\theta}$ term from K for the moment thus

$$\begin{aligned} K =& \sqrt{\frac{4}{3}\pi N} + (5 \times 10^{-8})K + 0.01K \\ \left(1 - (5 \times 10^{-8}) - 0.01\right) K =& \sqrt{\frac{4}{3}\pi N} \\ K \approx& 2.067327\sqrt{N}, \end{aligned}$$

and the result follows. □

4.1.7.1 Comparison

The expected running time of the three algorithms, Pollard's kangaroo, the method of van Oorschot & Wiener and the original Gaudry-Schost, are all de-

pendent on the position of Q. These running times are all symmetrical about the centre of the interval $[-N_1, N_1]$. To get an idea of how these algorithms compare we compare the average expected running time in Table 4.2, the worst case expected running time in Table 4.3 and the running times of different positions of Q in Figure 4.2. For the Gaudry-Schost algorithm in both the original and improved cases we take into account the bad steps to give a better comparison. We can see that the improved Gaudry-Schost algorithm is better than the original Gaudry-Schost algorithm in both the average and worst cases. We can see that

Name of Algorithm	Average Expected Running time
Van Oorschot & Wiener optimised for the average case	$2(1+\epsilon)\sqrt{N}+\frac{1}{\theta}$
Van Oorschot & Wiener optimised for the worst case	$2.16(1+\epsilon)\sqrt{N}+\frac{1}{\theta}$
Original Gaudry-Schost	$2.1\sqrt{N}+\frac{1}{\theta}$
Improved Gaudry-Schost	$2.07\sqrt{N}+\frac{1}{\theta}$

Table 4.2: Average expected running time of algorithms solving the DLP in an interval of size N

Name of Algorithm	Worst Case Expected Running Time
Van Oorschot & Wiener optimised for the average case	$3(1+\epsilon)\sqrt{N}+\frac{1}{\theta}$
Van Oorschot & Wiener optimised for the worst case	$2.83(1+\epsilon)\sqrt{N}+\frac{1}{\theta}$
Original Gaudry-Schost	$2.53\sqrt{N}+\frac{1}{\theta}$
Improved Gaudry-Schost	$2.07\sqrt{N}+\frac{1}{\theta}$

Table 4.3: Worst case expected running time of algorithms solving the DLP in an interval of size N

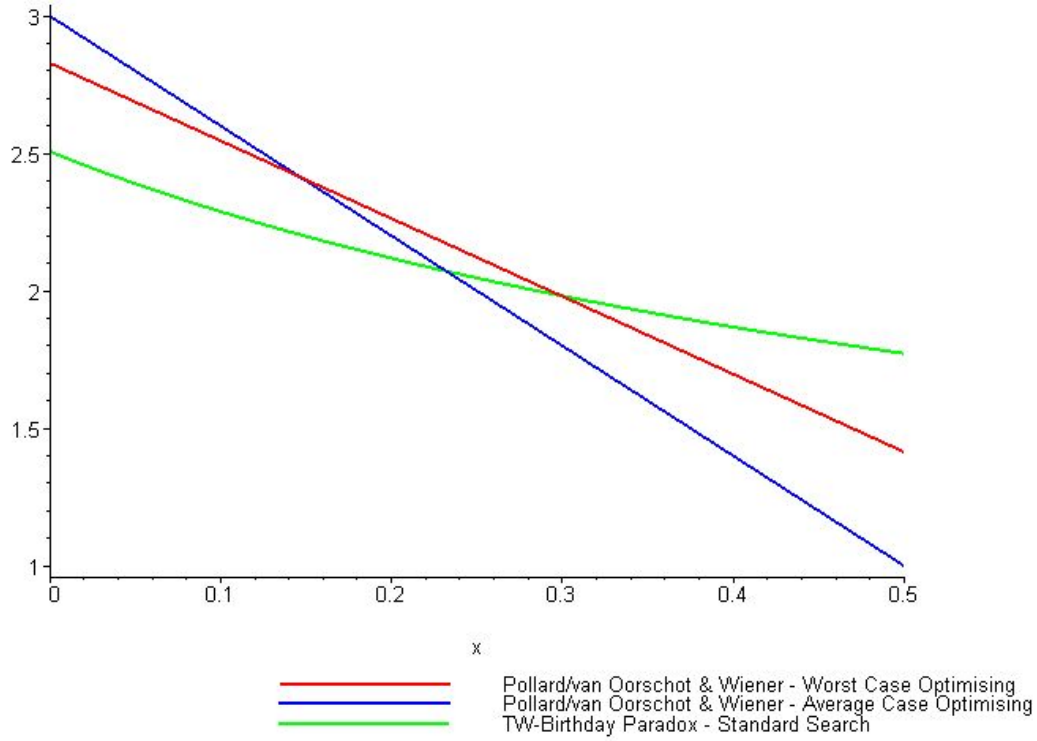

Figure 4.2: Running time as $Q = [-N_1 + xN]P$ for $0 \leq x \leq 1/2$

the improved Gaudry-Schost algorithm is slightly slower that the method by van Oorschot & Wiener but is still competitive. In addition the Gaudry-Schost algorithm does not require knowledge of the number of processors prior to starting and so, unlike the method of van Oorschot and Wiener, processors can be added and removed while the algorithm is running.

N.B. In the remaining sections of this chapter we focus on average expected running times.

4.2 Pollard's improved kangaroo algorithm

Very recently (and as a result of a personal communication with us) Pollard has improved the running time of the kangaroo algorithm by using more than 2 kangaroos [13, 36]. We first describe the intuition behind this improvement and then give the improved running time analysis. The van Oorschot and Wiener method which parallelises the original kangaroo algorithm and thereby having a probability of success close to, if not equal to, 1 can also be applied to Pollard's improved algorithms. For simplicity we explain the improvement in terms of the serial case and omit the extra steps with regards to distinguished points in Theorems 4.2.3 and 4.2.4. However in Section 4.2.3 we do briefly discuss the van Oorschot and Wiener version of Pollard's improved kangaroo algorithms.

The intuition behind this improvement is as follows. Using the notation of Definition 3.0.9 let us assume that we can calculate the inverse of Q i.e. $-Q = [-n_1]P_1$. In the case of elliptic curves this is trivial. Now let it be the case that we have started two wild walks, one at Q and the other at $-Q$ which have collided so

$$\begin{aligned} Q + [a_{w1}]P_1 &= -Q + [a_{w2}]P_1 \\ n_1 + a_{w1} &= -n_1 + a_{w2} \\ n_1 &= (a_{w2} - a_{w1})/2. \end{aligned}$$

Therefore we can solve for the discrete logarithm of Q. This shows that a WW collision of a certain type can be useful. Pollard used this idea to show that 3 kangaroos or even 4 kangaroos walking a smaller distance in terms of the interval of exponents can solve the DLP in an interval in less steps that the standard 2

kangaroo approach. We will briefly describe the algorithm as well as the improved running time.

4.2.1 The 3 kangaroo improvement

Given a problem instance without loss of generality let $Q = [n_1]P_1$ where $n_1 \geq 0$. Q and its inverse $-Q$ are equidistant from the centre of the interval so as Q gets further away from the centre so does $-Q$.

Definition 4.2.1.

- Let the walk starting at $-Q$ be called the Wild Negative or 'WildN' walk.
- Let the walk starting at Q be called the Wild Positive or 'WildP' walk.
- Let the starting point of the Tame walk be denoted R_t.

Definition 4.2.2. The different collisions between the walks will be defined as follows: -

- A **TWN collision** is a collision resulting from a Tame and WildN walk terminating at the same distinguished point.
- A **TWP collision** is a collision resulting from a Tame and WildP walk terminating at the same distinguished point.
- A **WNWP collision** is a collision resulting from a WildN and WildP walk terminating at the same distinguished point.

We will have two wild walks and a tame walk as given in Definition 4.2.1. In the 3 kangaroo case the collisions defined in Definition 4.2.2 are all useful in solving

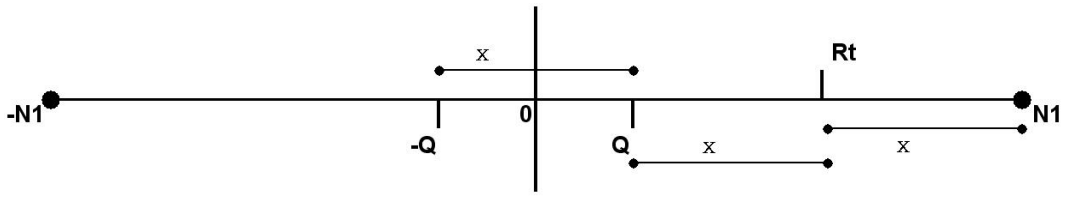

Figure 4.3: Choosing the starting point of the Tame walk

the DLP in an interval. Finally we need to decide the value of R_t. To reduce the running time of the kangaroo algorithm the tame walk does not start in the middle of the interval but, without loss of generality, at some point in the positive half of the interval. Figure 4.3 shows the situation where Q is equidistant from $-Q$ and R_t. This is where a TWP and a WNWP collisions are equally likely. The probability of a TWP collision should be equally likely when the problem instance is the same distance x to the left and right of R_t i.e. $Q = [R_t - x]P_1$ and $Q = [R_t + x]P_1 = [N_1]P_1$ as shown in Figure 4.3. Therefore we can solve equation (4.6) in x to obtain $x = 2N_1/5$ so $R_t = [\lfloor 3N_1/5 \rfloor]P_1$.

$$N_1 = x/2 + x + x. \tag{4.6}$$

Theorem 4.2.3. *Using Pollard's improved 3 kangaroo algorithm in the serial case to solve the DLP in an interval of size N and assuming the analogous Heuristic 3.1.1 holds, if we optimise for the average case then the mean step size is $m = \sqrt{N/10}$ and the average case running time is at most $1.9(1+\epsilon)\sqrt{N}$ group operations (excluding the time taken to invert Q).*

Proof. Effectively we can view the 3 kangaroo algorithm as the original algorithm working on a smaller interval but running 3 kangaroos instead of 2. The tame walk starts in the middle of this smaller interval. Therefore the size of the new

smaller interval is $4N_1/5 \approx 2N/5$. From part 2 of Theorem 3.1.2 the previous mean step size in the average case was $\sqrt{N}/2$ so the new mean step size is given by

$$m = \frac{1}{2} \cdot \sqrt{\frac{2N}{5}} = \sqrt{\frac{N}{10}}.$$

Using a similar technique with the running time from part 2 of Theorem 3.1.2 we obtain the following number of group operations in the average case

$$\frac{3}{2} \cdot 2(1+\epsilon)\sqrt{\frac{2N}{5}} = 3\sqrt{2}(1+\epsilon)\sqrt{\frac{N}{5}} \approx 1.9(1+\epsilon)\sqrt{N}.$$

□

The running time given in Theorem 4.2.3 is in fact pessimistic. The reason for this is that the analysis for Pollard's kangaroo algorithm does not take into account the possibility of a third type of collision. For a problem instance where n_1 is close to 0 there is a possibility of a TWN collision as well. So when this algorithm is put into practice we can expect to achieve an even faster running time than those given above. It remains an open problem to give an exact analysis of this algorithm. Nonetheless this running time is an improvement on the original kangaroo algorithm and in the next section we will see Pollard's 4 kangaroo improvement.

4.2.2 The 4 kangaroo improvement

The 4 kangaroo algorithm improves upon the 3 kangaroo algorithm by altering the pseudorandom walk and by having another tame walk. As Q and $-Q$ are equidistant from the centre of the interval of exponents n_1 and $-n_1$ take the

same value modulo 2 and are always an even distance apart. Therefore if the pseudorandom walks only takes even steps then the exponents of all footprints will either be all even or all odd: but we do not know which. This effectively reduces the size of the search space by half. However if n_1 is odd the tame walk will never collide with a wild walk. To resolve this Pollard used two tame walks which start next to each other i.e. on elements with exponents $\lfloor 3N_1/5 \rfloor$ and $\lfloor 3N_1/5 \rfloor + 1$. This ensures an equal number of even and odd footprints. Although half of the tame steps are immediately wasted there is still an improvement as detailed in the following theorem.

Theorem 4.2.4. *Using Pollard's improved 4 kangaroo algorithm to solve the DLP in an interval of size N and assuming the analogous Heuristic 3.1.1 holds, if we optimise for the average case then the mean step size is $m = \sqrt{N/10}$ and the average case running time is at most $1.79(1+\epsilon)\sqrt{N}$ group operations (excluding the time taken to invert Q).*

Proof. Following on from Theorem 4.2.3 the mean step size will remain at $\sqrt{N/10}$ except that now all steps will be even. We can look at the search space as being half its original size as we are either looking at all the even or odd exponents but to counteract this we are now doing 4 walks instead of 3. Therefore using the result from Theorem 4.2.3 the average expected running time is given by

$$\frac{4}{3} \cdot \frac{1}{\sqrt{2}} \cdot 3\sqrt{2}(1+\epsilon)\sqrt{\frac{N}{5}} = 1.79(1+\epsilon)\sqrt{N},$$

group operations. □

4.2.3 The Distributed algorithm and Comparison

The method of van Oorschot and Wiener which was applied to the original kangaroo algorithm can also be easily applied to the improved kangaroo algorithm i.e. there will be herds of tame, wildp and wildn walks and every time a walk lands on a distinguished point it is stored on a server in an easily accessible data structure. There is a linear speed up (by the number of processors) to the running times in Theorems 4.2.3 and 4.2.4 and an additional term of $1/\theta$ added on. As we have explained above the running times given in Theorems 4.2.3 and 4.2.4 are pessimistic. Pollard carried out some simulations on the 3 and 4 kangaroo variants which show that this is the case. The results of these are given in Section 4.4. Table 4.4 gives a breakdown of the running time (in group operations in the serial case) of the current algorithms to solve the DLP in an interval of size N. For simplicity we omit the $1/\theta$ terms, take $\epsilon = 0$, exclude the bad steps in the Gaudry-Schost algorithms and also we assume that the time required to find the inverse of Q is negligible compared to the overall running times.

Name of Algorithm	Average Expected Running time
Van Oorschot & Wiener optimised for the average case	$2\sqrt{N}$
Van Oorschot & Wiener optimised for the worst case	$2.16\sqrt{N}$
Original Gaudry-Schost	$2.08\sqrt{N}$
Improved Gaudry-Schost	$2.05\sqrt{N}$
Pollard 3 kangaroo	$\leq 1.90\sqrt{N}$
Pollard 4 kangaroo	$\leq 1.79\sqrt{N}$.

Table 4.4: Expected running time of algorithms solving the DLP in an interval of size N (Update 1)

An obvious question at this point is; Can we implement the techniques that Pollard used to speed up his kangaroo algorithm to the original and/or improved Gaudry-Schost algorithm? We answer this question in the next section.

4.3 Further improvements to the Gaudry-Schost algorithm

In this section we will apply Pollard's techniques to both the original and improved Gaudry-Schost algorithms to obtain faster running times for these algorithms.

4.3.1 The 3 set Gaudry-Schost algorithm

As before given a problem instance, without loss of generality, let $Q = [n_1]P_1$ where $n_1 \geq 0$. Using the calculations from Section 4.2.1, to improve the original Gaudry-Schost algorithm, denoted the 3 set Gaudry-Schost algorithm, we redefine the previous tame and wild sets as described below.

Definition 4.3.1. Given the DLP in an interval as in Definition 3.0.9 we define the following sets below: -

$$\begin{aligned} &\text{Tame Set} \quad T = \{a_1 \mid a_1 \in [N_1/5, N_1]\}, \\ &\text{WildN set} \quad W_N = -n_1 - 3N_1/5 + T, \\ &\text{WildP set} \quad W_P = n_1 - 3N_1/5 + T. \end{aligned}$$

We can picture these sets as shown in Figure 4.4. We apply the original Gaudry-Schost algorithm in this new setting by splitting the number of processors, N_P,

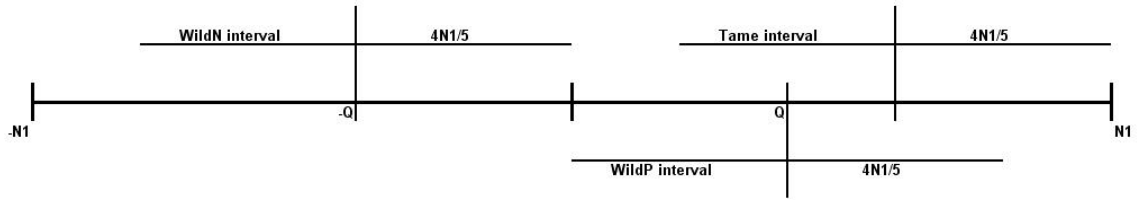

Figure 4.4: The Tame, WildN and WildP Sets

equally between walking in the tame, wildn and wildp sets. As in the 3 kangaroo algorithm either a TWP or WNWP collision is sufficient to solve the DLP in an interval. Also as n_1 approaches 0 there is also a nonzero probability of a TWN collision which can also be used to solve the DLP in an interval. In order to apply the Tame-Wild Birthday Paradox we ignore such collisions so just as in the improved kangaroo algorithm the expected running time will be pessimistic. An open problem is to include this in the analysis but this would need different tools from probability.

As the density of walks in all sets will be the same we can treat the tame and wildn sets as one set which overlaps with the wildp set.

Definition 4.3.2. Define $\mathcal{A} = (T \cup W_N) \cap W_P$ to be the total overlap between the tame and wildp sets and the wildn and wildp sets. Let the cardinality of this overlap be given by $M = |\mathcal{A}|$.

Theorem 4.3.3. *Assuming Heuristic 4.1.4 the expected number of group operations before we can solve for the DLP in an interval of size N using the 3 set Gaudry-Schost algorithm (excluding the time taken to invert Q and the bad steps) is at most*

$$1.85\sqrt{N} + \frac{1}{\theta}.$$

Proof. The first thing to note when solving the DLP in an interval is that both

the kangaroo and the 3 set Gaudry-Schost algorithm solve for both the discrete logarithm of Q and $-Q$ at the same time. Therefore we only need to average over problem instances where $n_1 \geq 0$ (without loss of generality). Let the problem instance be written as $Q = [xN_1]P_1$ for $0 \leq x \leq 1$. Looking at Figure 4.4 we can calculate the expected number of group operations by taking the sum of averages scaled over smaller ranges of x.

1. Firstly let us consider those problem instances where $3/5 < x \leq 1$. Here $|W_N \cap W_P| = 0$ and $|T \cap W_N| = 0$. Therefore $M = (7/5 - x)N_1$ and the expected number of walks before we step in $\mathcal{A}$ is given by

$$\frac{\frac{4}{5}N_1}{\left(\frac{7}{5} - x\right)N_1} = \frac{4}{5}\left(\frac{7}{5} - x\right)^{-1}.$$

   So for a given problem instance in this range of x the expected number of group operations before a TWP or WNWP collision is

$$\frac{4}{5}\left(\frac{7}{5} - x\right)^{-1/2}\sqrt{\pi N_1}$$

   So averaging out over this range of x we must multiply the number of steps by $3/2$ as one third of the steps are wasted in W_N. Therefore over this range of x the expected number of group operations is given by

$$\frac{3}{2} \cdot \frac{4}{5}\sqrt{\pi N_1}\int_{x=3/5}^{1}\left(\frac{7}{5} - x\right)^{-1/2} dx = \frac{24 - 12\sqrt{2}}{5\sqrt{5}}\sqrt{\pi N_1}. \qquad (4.7)$$

2. For $2/5 < x \leq 3/5$ the overlap $T \cap W_P$ starts to decrease in size from the maximum but we still have $|T \cap W_N| = |W_N \cap W_P| = 0$ so the expected

number of walks before we step in $\mathcal{A}$ is given by

$$\frac{4}{5}\left(\frac{1}{5}+x\right)^{-1}.$$

Again we have to multiply the number of steps by 3/2 to take into account the steps wasted in W_N, so the expected number of group operations over this range of x is

$$\frac{3}{2}\cdot\frac{4}{5}\sqrt{\pi N_1}\int_{x=2/5}^{3/5}\left(\frac{1}{5}+x\right)^{-1/2}dx=\frac{24-12\sqrt{3}}{5\sqrt{5}}\sqrt{\pi N_1}. \tag{4.8}$$

3. For $1/5 < x \leq 2/5$ both $T \cap W_P$ and $W_N \cap W_P$ are non empty. To continue using the Tame-Wild Birthday Paradox to model this problem we treat $T \cup W_N$ as one set. Although compared to W_P there will be twice as many walks in $T \cup W_N$ this set is also twice as large so the densities of footprints in $T \cup W_N$ and W_P is the same. Therefore the cardinality of $\mathcal{A}$ is given by

$$M = 2\left(\frac{2}{5}-x\right)N_1+\left(\frac{1}{5}+x\right)N_1=(1-x)\,N_1.$$

So the number of steps in W_P before we expect a TWP or WNWP collision is given by

$$\frac{4}{5}\,(1-x)^{-1/2}\,\frac{\sqrt{\pi N_1}}{2}.$$

Remember that for every 1 step in W_P we are doing 2 steps $T \cup W_N$. So the number of steps in this union before we expect a TWP or WNWP collision is given by

$$\frac{8}{5}\,(1-x)^{-1/2}\,\frac{\sqrt{\pi N_1}}{2}.$$

Put these together and integrating over the range of x we obtain the following number of group operations

$$\frac{6}{5}\sqrt{\pi N_1}\int_{x=1/5}^{2/5}(1-x)^{-1/2}\,dx = \frac{24-12\sqrt{3}}{5\sqrt{5}}\sqrt{\pi N_1}. \tag{4.9}$$

4. Finally for $0 \leq x \leq 1/5$ the overlap $T \cap W_N$ is nonempty as well so there is a possibility of a TWN collision. In fact when looking at the set $T \cap W_N$ we see that there is a double density of walks when compared with W_P. In order to use the Tame-Wild Birthday Paradox we were not able to model this (it remains an open problem to include different densities in the analysis). Therefore we treat all problem instances in this range of x such that $T \cap W_N$ is empty and that $|T \cup W_N| = 8N_1/5$. This means that in a similar way to the improved Pollard kangaroo algorithm our result will be pessimistic. For all problem instances in this range of x, $\mathcal{A} = W_P$ so the number of steps in W_P before we expect a collision is

$$\frac{\sqrt{\pi 4N_1/5}}{2} = \sqrt{\frac{\pi N_1}{5}}.$$

The expected number of steps in $T \cap W_N$ is twice the above as we expect to do 2 walks before we get a walk in $\mathcal{A}$. So the expected number of group operations on average scaled over this range of x is at most

$$\frac{1}{5}\cdot\frac{3\sqrt{\pi 4N_1/5}}{2} = \frac{3\sqrt{\pi N_1}}{5\sqrt{5}}. \tag{4.10}$$

Adding together the results from equations (4.7), (4.8), (4.9) and (4.10) gives the expected number of group operations to solve the DLP in the interval of size N

which is

$$\frac{75 - 12\sqrt{2} - 24\sqrt{3}}{5\sqrt{5}}\sqrt{\pi N_1} \approx 1.85\sqrt{N}.$$

Once a collision has occurred it is only identified when both walks hit a distinguished point and the result follows. □

4.3.2 The 4 set Gaudry-Schost algorithm

In the same way as the 4 kangaroo algorithm we can improve the 3 set Gaudry-Schost algorithm by making the following changes to obtain a new algorithm which we will call the 4 set Gaudry-Schost algorithm: -

1. Making the pseudorandom walk only take steps of even size,
2. Having twice as many tame walks as either wildn and wildp walks, half searching over even exponents and half over odd exponents,
3. Starting all wildn walks with the form: $y_1 = -Q + [b_1]P_1$ where $b_1 \equiv 0 \mod 2$,
4. Starting all wildp walks with the form: $z_1 = Q + [c_1]P_1$ where $c_1 \equiv 0 \mod 2$.

Theorem 4.3.4. *Assuming Heuristic 4.1.4 the expected number of group operations before we can solve for the DLP in an interval of size N using the 4 set Gaudry-Schost algorithm (excluding the time taken to invert Q and the bad steps) is at most*

$$1.74\sqrt{N} + \frac{1}{\theta}.$$

Proof. Taking the result from Theorem 4.3.3 the interval is now effectively halved in size as we are only taking even steps but we have an additional set of tame

walks which only have footprints with odd discrete logarithms in case n_1 is odd. Therefore the expected number of group operations for the 4 set Gaudry-Schost algorithm to solve the DLP in an interval of size N is given by

$$\frac{4}{3} \cdot \frac{1}{\sqrt{2}} \cdot 1.85\sqrt{N} + \frac{1}{\theta} \approx 1.74\sqrt{N} + \frac{1}{\theta}.$$

□

4.3.3 The improved 3 and 4 set Gaudry-Schost algorithm

Theorem 4.3.4 is an improvement on the 4 kangaroo algorithm but we can go further. By employing Pollard's 4 kangaroo technique on the improved Gaudry-Schost algorithm we obtain an even faster algorithm which we call the improved 4 set Gaudry-Schost algorithm. This algorithm works in exactly the same way as the 4 set Gaudry-Schost algorithm except that we search only the central two thirds of the tame set and the same sized search area but split equally between the extreme ends of the wildn and wildp sets. Figure 4.5 shows these new search areas as shaded regions. The first diagram in Figure 4.5 shows a best case where $Q = [4N_1/15]P_1$ and the second diagram shows the only worst case where $Q = [2N_1/15]P_1$.

The new search areas are defined in Definition 4.3.5.

Definition 4.3.5. Given the DLP in an interval as in Definition 3.0.9 we define the following sets below: -

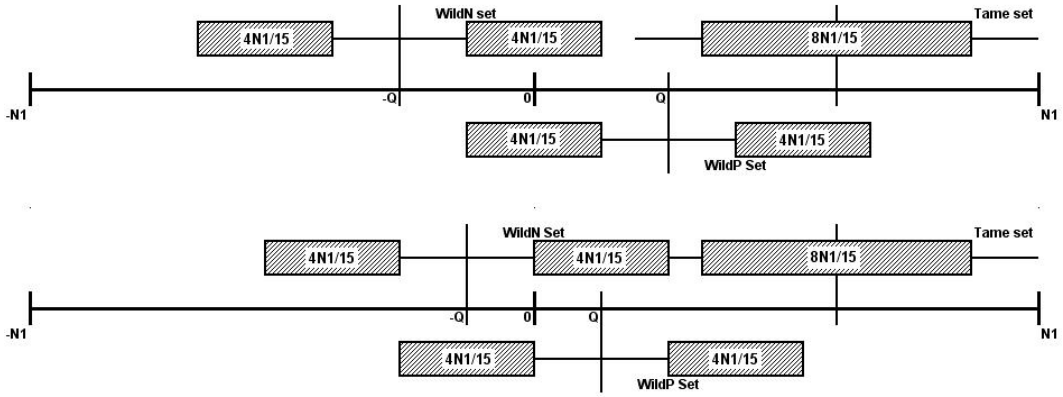

Figure 4.5: The new Tame, WildN and WildP Sets in a best and worst case problem instance

$$\begin{aligned}
&\text{New Tame Set} \quad T' = \{a_1 \mid a_1 \in [N_1/3, 13N_1/15]\},\\
&\text{New WildN set} \quad W'_P = (n_1 - 2N_1/5 + \mathcal{S}) \cup (n_1 + 2N_1/15 + \mathcal{S}),\\
&\text{New WildP set} \quad W'_N = (-n_1 - 2N_1/5 + \mathcal{S}) \cup (-n_1 + 2N_1/15 + \mathcal{S}).
\end{aligned}$$

where $\mathcal{S} = \{a_1 \mid a_1 \in [0, 4N_1/15]\}$

Definition 4.3.6. Define $\mathcal{A}' = (T' \cup W'_N) \cap W'_P$ to be the total overlap between the new tame and wildp sets and the new wildn and wildp sets. Let the cardinality of this overlap be given by $M' = |\mathcal{A}'|$.

Theorem 4.3.7. *Assuming Heuristic 4.1.4 the expected number of group operations before we can solve for the DLP in an interval of size N using the improved 4 set Gaudry-Schost algorithm (excluding the time taken to invert Q and the bad steps) is*

$$1.72\sqrt{N} + \frac{1}{\theta}.$$

Proof. We prove this lemma by first calculating the expected number of group operations for the improved 3 set Gaudry-Schost algorithm to solve the DLP in an interval of size N then consider the 4 set improvements. As before, the

algorithm works on both Q and $-Q$ at the same time we only need to average over problem instances where $n_1 \geq 0$, without loss of generality. Therefore let the problem instance be written as $Q = [xN_1]P_1$ for $0 \leq x \leq 1$. Looking at Figure 4.5 we can calculate the expected number of group operations by taking the sum of averages scaled over smaller ranges of x.

1. For $2/5 < x \leq 1$ we have seen in Section 4.1.5 that the overlap is $\mathcal{A}' = 4N_1/15$. All walks in W'_N are wasted and in T' and W'_P we expect to do two walks before we are walking in the overlap so the expected number of group operations scaled over this range of x is given by

$$\frac{3}{5} \cdot \frac{3}{2} \cdot 2\sqrt{\pi \frac{4N_1}{15}} = \frac{18}{5\sqrt{15}}\sqrt{\pi N_1}. \tag{4.11}$$

2. For $4/15 < x \leq 2/5$ the cardinality of $\mathcal{A}'$ increases as $W'_N \cap W'_P$ is now nonempty. So the overlap is given by

$$M' = 2\left(\frac{2}{5} - x\right)N_1 + \frac{4N_1}{15} = \left(\frac{16}{15} - 2x\right)N_1.$$

   As both $T' \cap W'_P$ and $W'_N \cap W'_P$ are nonempty we model $T' \cup W'_N$ as one set which although is twice as large as W'_P there are twice as many walks in it so the density of the walks is the same. This enables us to use the Tame-Wild Birthday Paradox and so the expected number of steps in W'_P before a collision is given by

$$\frac{8}{15}\left(\frac{16}{15} - 2x\right)^{-1/2} \frac{\sqrt{\pi N_1}}{2}$$

and the expected number of steps in $T' \cup W'_N$ before a collision is double this so the expected number of group operations before a TWP or a WNWP collision scaled over this range of x is

$$\frac{12}{15}\sqrt{\pi N_1}\int_{4/15}^{2/5}\left(\frac{16}{15}-2x\right)^{-1/2}dx = \frac{8\sqrt{2}-8}{5\sqrt{15}}\sqrt{\pi N_1}. \tag{4.12}$$

3. For $1/5 < x \leq 4/15$ the overlap decreases from the maximum of $8N_1/15$ to $2N_1/5$ but the right half of W'_P is still contained in T. So the cardinality of $\mathcal{A}'$ is given by

$$M' = \frac{8N_1}{15} - 2\left(\frac{4}{15}-x\right)N_1 = 2xN_1.$$

Again we model $T' \cup W'_N$ as one set and so the expected number of steps in W'_P before a collision is given by

$$\frac{8}{15}(2x)^{-1/2}\frac{\sqrt{\pi N_1}}{2}$$

and the expected number of steps in $T' \cup W'_N$ before a collision is double this so the expected number of group operations before a TWP or a WNWP collision scaled over this range of x is

$$\frac{12}{15}\sqrt{\pi N_1}\int_{1/5}^{4/15}(2x)^{-1/2}\,dx = \frac{8\sqrt{2}-4\sqrt{6}}{5\sqrt{15}}\sqrt{\pi N_1}. \tag{4.13}$$

4. For $2/15 < x \leq 1/5$ the cardinality of the overlap drops from $2N_1/5$ when $x = 1/5$ to its minimum size across all problem instances. This is where

$W'_P \cap W'_N = \phi$. So the overlap is given by

$$M' = \left(x + \frac{1}{15}\right) N_1 + 2\left(x - \frac{2}{15}\right) N_1 = \left(3x - \frac{1}{5}\right) N_1.$$

Again we model $T' \cup W'_N$ as one set and so the expected number of steps in W'_P before a collision is given by

$$\frac{8}{15}\left(3x - \frac{1}{5}\right)^{-1/2} \frac{\sqrt{\pi N_1}}{2}$$

and the expected number of steps in $T' \cup W'_N$ before a collision is double this so the expected number of group operations before a TWP or a WNWP collision scaled over this range of x is

$$\frac{12}{15}\sqrt{\pi N_1} \int_{2/15}^{1/5} \left(3x - \frac{1}{5}\right)^{-1/2} dx = \frac{8\sqrt{2} - 8}{15\sqrt{5}} \sqrt{\pi N_1}. \tag{4.14}$$

5. For $1/15 < x \leq 2/15$ the size of the overlap increases as $W'_P \cap W'_N$ becomes nonempty again. The cardinality of $\mathcal{A}'$ is given by

$$M' = 3\left(\frac{2}{15} - x\right) N_1 + \frac{N_1}{5} = \left(\frac{3}{5} - 3x\right) N_1.$$

Again we model $T' \cup W'_N$ as one set and so the expected number of steps in W'_P before a collision is given by

$$\frac{8}{15}\left(\frac{3}{5} - 3x\right)^{-1/2} \frac{\sqrt{\pi N_1}}{2}$$

and the expected number of steps in $T' \cup W'_N$ before a collision is double this

so the expected number of group operations before a TWP or a WNWP collision scaled over this range of x is

$$\frac{12}{15}\sqrt{\pi N_1}\int_{1/15}^{2/15}\left(\frac{3}{5}-3x\right)^{-1/2}dx=\frac{8\sqrt{2}-8}{15\sqrt{5}}\sqrt{\pi N_1}. \tag{4.15}$$

6. Finally for $0 \leq x \leq 1/15$ $T' \cap W'_N$ is non empty so we have a double density of walks in this set which the Tame-Wild Birthday Paradox is not able to model. So in the same way as Theorem 4.3.3 we have to be pessimistic and pretend that this intersection is empty. So we treat the cardinality of this set as $|T'| + |W'_N| = 16N_1/15$. In this case we have

$$M' = \frac{2N_1}{5} + 2\left(\frac{1}{15}-x\right)N_1 = \left(\frac{8}{15}-2x\right)N_1.$$

The expected number of steps in W'_P before a collision is given by

$$\frac{8}{15}\left(\frac{8}{15}-2x\right)^{-1/2}\frac{\sqrt{\pi N_1}}{2}$$

and as we are modeling $T' \cup W'_N$ as one set such that $|T' \cap W'_N| = 0$ the expected number of steps in this set is double the above. Therefore the expected number of group operations before a TWP or a WNWP collision scaled over this range of x is

$$\frac{12}{15}\sqrt{\pi N_1}\int_{0}^{1/15}\left(\frac{8}{15}-2x\right)^{-1/2}dx=\frac{8\sqrt{2}-4\sqrt{6}}{5\sqrt{15}}\sqrt{\pi N_1}. \tag{4.16}$$

Adding together the results from equations (4.11), (4.12), (4.13), (4.14), (4.15) and (4.16) gives the expected number of group operations to solve the DLP in

the interval of size N using the improved 3 set Gaudry-Schost algorithm, which is

$$\left(\frac{10+24\sqrt{2}-8\sqrt{6}}{5\sqrt{15}}+\frac{16\sqrt{2}-16}{15\sqrt{5}}\right)\sqrt{\pi N_1}\approx 1.82\sqrt{N}.$$

Using the same technique as in Theorem 4.3.4, we employ a walk which only makes steps of even size but we also have double the number of tame walks. Then the the expected number of group operations to solve the DLP in the interval of size N using the improved 4 set Gaudry-Schost algorithm is approximately

$$\frac{4}{3}\cdot\frac{1}{\sqrt{2}}\cdot 1.82\sqrt{N}\approx 1.72\sqrt{N}.$$

Once a collision has occurred it is only identified when both walks hit a distinguished point and the result follows. □

4.4 Comparison and Experimental results

Table 4.5 gives the expected running times (with $\epsilon = 0$, without counting bad steps and omitting the $1/\theta$ terms) of the different algorithms for solving the DLP in an interval of size N. The improved 4 set Gaudry-Schost is currently the fastest algorithm for solving this problem. However the difference in running times between the Pollard 4 kangaroo, 4 set and improved 4 set Gaudry-Schost is very small. There is only a difference of 0.07 between the constants. Therefore to really compare them we have to consider experimental data. In Pollards personal communication with us [36] his experiments produced an average running time of $1.75\sqrt{N}$ and with some tweaks in the mean step size as low as $1.71\sqrt{N}$ so let us see how the Gaudry-Schost algorithms fare.

Name of Algorithm	Expected Running time
Van Oorschot & Wiener optimised for the average case	$2\sqrt{N}$
Van Oorschot & Wiener optimised for the worst case	$2.16\sqrt{N}$
Original Gaudry-Schost	$2.08\sqrt{N}$
Improved Gaudry-Schost	$2.05\sqrt{N}$
Pollard 3 kangaroo	$\leq 1.90\sqrt{N}$
Pollard 4 kangaroo	$\leq 1.79\sqrt{N}$
3 set Gaudry-Schost	$\leq 1.85\sqrt{N}$
4 set Gaudry-Schost	$\leq 1.74\sqrt{N}$
Improved 4 set Gaudry-Schost	$\leq 1.72\sqrt{N}$.

Table 4.5: Expected running times of algorithms solving the DLP in an interval of size N (Update 2)

Using Magma we simulated a group and implemented the 3 set, 4 set and improved 4 set Gaudry-Schost algorithms. Experiments were done on three separate interval sizes which were approximately 2^{34}, 2^{40} and 2^{47} in size. The average running times for the different experiments and algorithms are given in Table 4.6. Walks were not permitted to start m/θ from the edge of any of the sets. However we found that the chance of a walk starting close to this new edge was very small. In fact no walks stepped outside of the tame and wild sets. Therefore to achieve a better coverage of these sets we reduced the size of the subsets where walks were not allowed to start to approximately $m/2\theta$.

In Experiment 1 all the algorithms produced a worse running time that those given in Table 4.5. A possible reason for this is that the interval was too small for the pseudorandom walk to be random enough to apply the Tame-Wild Birthday Paradox accurately. However as the interval sizes increased in Experiment 2 and

3, we attained running times even faster than those presented in the theoretical results. This, we believe, is due to the fact that the theoretical results are pessimistic as they do not take in account the 3-set overlap $T \cap W_N \cap W_P$ being nonempty when problem instances approaches $Q = [0]P_1$. In addition our experimental results give further evidence that the 4 set Gaudry-Schost algorithm is faster than the 3 set variant and the improved 4 set variant is faster still.

	# of Experiments	3 set	4 set	Improved 4 set
Experiment 1 $N \approx 2^{34}$ $m/\theta \approx 2^{15}$	1000	$1.89\sqrt{N}$	$1.79\sqrt{N}$	$1.83\sqrt{N}$
Experiment 2 $N \approx 2^{40}$ $m/\theta \approx 2^{20}$	300	$1.77\sqrt{N}$	$1.74\sqrt{N}$	$1.71\sqrt{N}$
Experiment 3 $N \approx 2^{48}$ $m/\theta \approx 2^{24}$	50	$1.75\sqrt{N}$	$1.62\sqrt{N}$	$1.51\sqrt{N}$

Table 4.6: Average running times of Gaudry-Schost algorithms for solving the DLP in an interval of different sizes of N

Chapter 5

Computing Discrete Logarithms in Intervals using Equivalence Classes

Having already made improvements to the algorithms for solving the DLP in an interval, in this chapter we go further in specific groups. Let us now consider groups in which we have a fast inversion. What we mean by 'fast' is that the time taken to compute inverses is negligible. An example of such a group is the set of points on an elliptic curve. E.g. Consider the elliptic curve $E : y^2 = x^3 + Ax + B$ over a finite field $\mathbb{F}_q$ where q is an odd prime. If we let $P = (x_P, y_P) \in E(\mathbb{F}_q)$ then the inverse of P is simply $-P = (x_P, -y_P)$.

We first present the equivalence class analogue of the original Gaudry-Schost algorithm in Section 5.2 then present an improvement to the original Gaudry-Schost algorithm in groups with fast inversion which utilises equivalence classes. This

improved Gaudry-Schost algorithm using equivalence classes is now the fastest algorithm for solving the DLP in an interval in groups with fast inversion. Our research paper on this topic [14] was accepted to the Indocrypt 2009 conference in New Delhi, India but has since been withdrawn and resubmitted to the Public Key Cryptography 2010 conference in Paris, France.

5.1 Pseudorandom walks on Equivalence Classes

We improve the Gaudry-Schost algorithm by considering sets of equivalence classes as opposed to just elements of the group. Following the work of Gallant, Lambert and Vanstone [17] and Wiener and Zuccherato [50] it is natural to consider a pseudorandom walk on a set of equivalence classes. For the DLP in an interval this only makes sense when the equivalence class is a set of group elements all of whose discrete logarithms lie in the interval. Groups with fast inversion are good candidates for this.

Let the DLP in an interval be defined as in Definition 3.0.9. An equivalence class consists of an element of the group together with its inverse. So given the problem instance Q the analogous equivalence class problem instance is $\{Q, -Q\}$. As the discrete logarithm of Q, n_1, belongs to the interval $[-N_1, N_1]$ so does $-n_1$, the discrete logarithm of $-Q$. It is necessary to be able to compute a unique representative of the equivalence class so that one can define a deterministic pseudorandom walk on the equivalence classes. A simple rule in the elliptic curve case when q is prime is: treat the y-coordinate of P as an integer $0 \leq y_P < q$ and let the unique representative be $(x_P, \min\{y_P, q - y_P\})$. The pseudorandom walk is then defined using the unique equivalence class representative.

If we denote elements of the group by their discrete logarithms and order those in the interval $[-N_1, N_1]$, then the two elements in an equivalence class are equidistant from the centre of the interval. The standard Gaudry-Schost pseudorandom walk (Pseudorandom walk 3), in this setting, consists of a set of different sized jumps uniformly distributed in $[0, 2m]$ where m is the mean step size. However due to this symmetry, we effectively have a side-to-side walk in the set of equiv-

alence classes which is taking steps in the interval $[-2m, 2m]$. As we will see in Figures 5.1 and 5.2 diagramatically and equations (5.3) and (5.6), walks in the equivalence classes are analogous to looking at walks on one side of the interval of exponents as there are only $1 + N/2$ equivalence classes.

Therefore we need a different method of determining the size of the subsets (one on each side) of the tame and wild sets where walks will not start. As we effectively have a side-to-side walk the size of the subsets where walks are not allowed to start will be the expected maximum distance/excursion of the walk from its starting point. We recall the result of Cofman, Flajolet, Flatto and Hofri [7].

Lemma 5.1.1. *Let $y_0, y_1, \ldots, y_k$ be a symmetric random walk that starts at the origin ($y_0 = 0$) and takes steps uniformly distributed in $[-1, +1]$ then the expected maximum excursion is*

$$E(\max\{|y_i| : 0 \leq i \leq k\}) = \sqrt{\frac{2k}{3\pi}} + O(1)$$

N.B. The mean absolute step size in this walk is $\frac{1}{2}$.

Again let m be the mean step size. Then the size of the subset where walks are not allowed to start will be

$$2m\sqrt{\frac{2}{3\pi\theta}}. \tag{5.1}$$

To get a rough idea of how big these subsets are in terms of N, in order to use only constant storage we must take $\theta = c/\sqrt{N}$ (as shown in Section 4.1.6). Then the expression in equation (6.10) becomes

$$2m\sqrt{\frac{2\sqrt{N}}{3\pi c}} = O(N^{1/4}). \tag{5.2}$$

This is very small in comparison to the size of the interval so in practice does not effect the overall running time of the algorithm.

However an important issue is that there is a danger of small cycles in the walks. This phenomena was noted by Gallant, Lambert and Vanstone [17] and Wiener and Zuccherato [50]. This would cause the pseudorandom walks never to reach a distinguished point. A method to get around this problem is called "collapsing the cycle" which can be found in Gallant, Lambert and Vanstone [17, Section 6]. Briefly, collapsing the cycle is a method that a pseudorandom walk can use to detect that it has entered into a cycle and at a specific element of the cycle the walk takes a step of special size to get out of the cycle. In order for the walk to remain deterministic both the method of the detecting cycles and stepping to leave the cycle must be done in a deterministic way.

Example 5.1.2. A trivial way to detect that a walk has entered a cycle is as follows: We can record a fixed number of previous footprints of the walk and at each step update this list as well as checking whether the current footprint is in the list. If it is, then a cycle has been found. There is a very small probability of larger cycles occurring so we only need to store a small number of previous footprints, say 10-30. However unless we store all the points in each walk there will also be a possibility that a cycle will be missed. When a cycle has been found we need to choose a unique element of the cycle. In the case of elliptic curves a simple rule could be to choose a point in the cycle with the largest x-coordinate. Then from that point the walk can take a step of a size specifically reserved for collapsing the cycle and the walk is now outside of the cycle whilst it is still deterministic.

Whilst Example 5.1.2 describes a method for collapsing the cycle which is quite inefficient the method described by Gallant, Lambert and Vanstone [17] has only a negligible effect on the overall running time of the algorithm. Also the method of Nivasch [32] for detecting small cycles in long sequences using a stack would be a better approach.

Having addressed and resolved the standard issues on equivalence classes we can apply the Gaudry-Schost algorithm on equivalence classes to solve the DLP in an interval of size N.

5.2 The Original Gaudry-Schost algorithm on Equivalence Classes

Recall from Definition 3.0.9 that we are trying to find the discrete logarithm n_1 of an element Q of the group.

Definition 5.2.1. Let us redefine the tame set T and the wild set W as sets of exponents of equivalence classes as follows

$$T = [\{a, -a\} \mid a \in [-N_1, N_1]],$$
$$W = [\{n_1 + a, -(n_1 + a)\} \mid a \in [-N_1, N_1]].$$

We will first consider the case where we are searching all of the tame and wild sets as given in Definition 5.2.1. Actually, as we will be using Pseudorandom

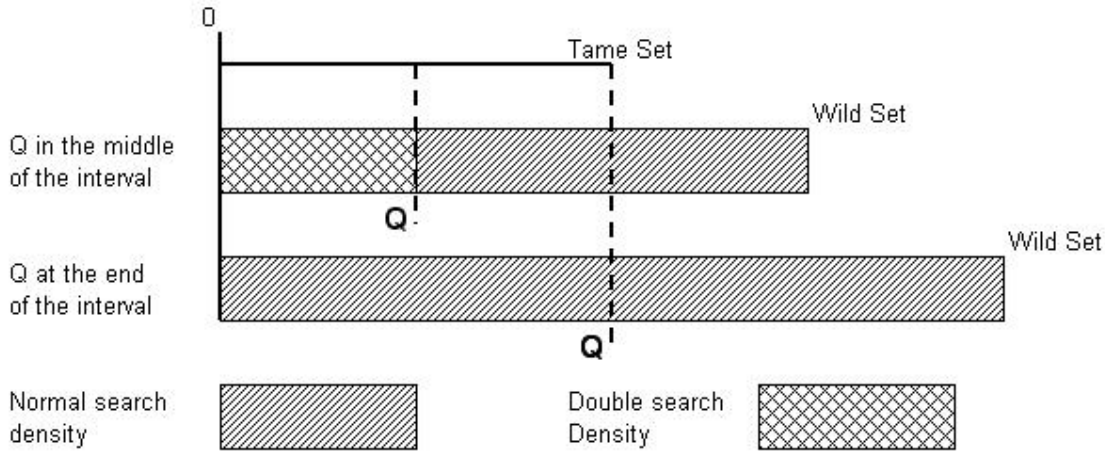

Figure 5.1: Searching equivalence classes: Q in the middle and at the end of the interval

walk 3, walks will start in the following subsets of T and W respectively

$$\text{Subset of } T = \left\{ \{a, -a\} \mid a \in \left[-N_1 + 2m\sqrt{\frac{2}{3\pi\theta}}, N_1 - 2m\sqrt{\frac{2}{3\pi\theta}} \right] \right\},$$

$$\text{Subset of } W = \left\{ \{n_1 + a, -(n_1 + a)\} \mid a \in \left[-N_1 + 2m\sqrt{\frac{2}{3\pi\theta}}, N_1 - 2m\sqrt{\frac{2}{3\pi\theta}} \right] \right\}.$$

As discussed in the last section, due to symmetry, the number of equivalence classes in the Tame set is $1 + N_1$ or $1 + N/2$ but in the worst case problem instance there are N equivalence classes in the wild set. To better understand what we are searching it is useful to consider the search areas visually. T has a 1-to-1 correspondence with the set $\{a \mid a \in [0, N_1]\}$ and is labeled the 'Tame set' in Figure 5.1. W is slightly more complicated to visualise due to the symmetry about the origin of the interval causing the set to change in size. Nevertheless W has a 1-to-1 correspondence with the set $\{|n_1| + a \mid a \in [-N_1, N_1]\}$ which in turn

has a 1-to-1 correspondence with the following multiset

$$A \cup B = \{|n_1| + a \mid a \in [-|n_1|, N_1]\} \cup \{-(|n_1| + a) \mid a \in [-N_1, -|n_1|]\}. \quad (5.3)$$

In fact $B \subseteq A$ and this is shown in Figure 5.1 by the cris-cross shading where we have a 'double density' of walks. One could argue that in this region there will be twice as many wild walks as there are tame walks. However this is not accurate as the same number of wild walks, as tame, are being spread over a larger search space. The Tame-Wild Birthday Paradox analysis does not take the different densities of walks/steps into account so it may be beneficial to have a different number of tame and wild walks.

Theorem 5.2.2. *Assuming Heuristic 4.1.4, if the proportion of walks in T and W is in the ratio $2:3$ then the expected number of group operations to solve the DLP in an interval of size N (without considering the bad steps) using the original Gaudry-Schost algorithm on equivalence classes is at most*

$$1.70\sqrt{N} + \frac{1}{\theta}$$

group operations.

Proof. Let $Q = [xN]P_1$ for $-1/2 \leq x \leq 1/2$. Without loss of generality we can consider the cases where x is positive then normalise to obtain the average expected running time. Firstly the intuition behind the ratio of pseudorandom walks in T and W being $2:3$ is the following. In the average case $|W| = 3|T|/2$ so in order for the densities of walks in both sets to be equal in the average case the ratio of walks in T and W must be $2:3$. However in all problem instances, except

when $x = 1/4$, there will be a double density of walks in a subset of W as shown in Figure 5.1. We model this situation using the Tame-Wild Birthday Paradox so we are unable to take the double density of walks into consideration. Hence, our results are pessimistic and in practice we expect the algorithm to require fewer steps than our results suggest. The overlap for all problem instances is given by $\mathcal{A} = T \cap W = T$. The cardinality of $\mathcal{A}$ is $1 + N_1 = 1 + N/2 \approx N/2$.

1. For problem instances where $0 \leq x \leq 1/4$ we will be doing more wild walks than what is required by the Tame-Wild Birthday Paradox. The number of steps in T before we expect a collision is $\sqrt{\pi N/2}/2$. Therefore, by the ratio of walks, the expected number of group operations scaled over this range of x is given by
$$\frac{1}{4} \cdot \frac{5\sqrt{\pi N/2}}{4}. \tag{5.4}$$

2. For $1/4 < x \leq 1/2$ the density of the tame walks will now be higher than that of the wild walks except in the region of the wild set which has double density. Again we will pretend that the extra walks in the region of the double density are wasted. The number of steps in W before we expect to step in the overlap $\mathcal{A}$ is given by $1+2x$, so the number of wild steps required in the overlap before we expect a collision is $(1+2x)\sqrt{\pi N/2}/2$. The number of tame steps will be $2/3$ of this making the total number of tame and wild steps $5(1 + 2x)\sqrt{\pi N/2}/6$. We need to integrate this expression over the range $1/4 < x \leq 1/2$ to obtain the expected number of group operations scaled over this range of x which is at most
$$\frac{5\sqrt{\pi N/2}}{6} \int_{x=1/4}^{1/2} (1 + 2x)dx = \frac{2.1875\sqrt{\pi N/2}}{6}. \tag{5.5}$$

Adding together the results from equations (5.4) and (5.5) and normalising the expected number of group operations before a collision is at most

$$2\left(\frac{5\sqrt{\pi N/2}}{16}+\frac{2.1875\sqrt{\pi N/2}}{6}\right)\approx 1.35\sqrt{\pi N/2}\approx 1.70\sqrt{N}.$$

Once a collision has occurred it is only identified when both walks hit a distinguished point and the result follows. □

This is a smaller expected running time when compared to the improved Pollard kangaroo and the improved 4 set Gaudry-Schost algorithm. In the next section we will improve on this running time by searching in a smaller region within the wild set.

5.3 The Improved Gaudry-Schost algorithm on Equivalence Classes

To improve on the original Gaudry-Schost algorithm on equivalence classes we search only a subset of the wild set such that in the worst case we are searching the same number of equivalence classes as in the tame set. To do this we only search half of the wild set centred in the middle whilst still searching all of the tame set. This subset of W is described below.

Definition 5.3.1. Let the 'refined' wild set be given by

$$W' = \{\{n_1 + a, -(n_1 + a)\} \mid a \in [-N_1/2, N_1/2]\}.$$

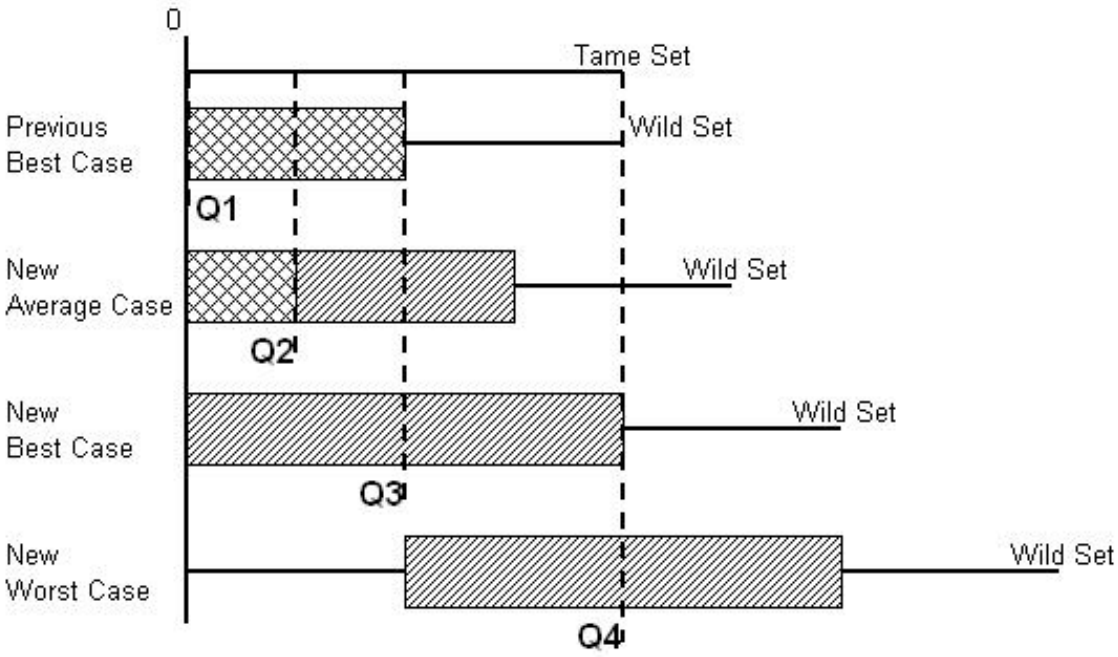

Figure 5.2: Searching only half of the Wild Set

As in the original Gaudry-Schost case to better understand the new search area we can consider it visually. W' has a 1-to-1 correspondence with $\{|n_1| + a \mid a \in [-N_1/2, N_1/2]\}$ for all possible n_1 but for $|n_1| \leq N_1/2$ the set W' also has a 1-to-1 correspondence with the following multiset

$$C \cup D = \{|n_1| + a \mid a \in [-|n_1|, N_1/2]\} \cup \{-(|n_1| + a) \mid a \in [-N_1/2, -|n_1|]\}. \quad (5.6)$$

Again $D \subseteq C$ and this is shown in Figure 5.2, for the problem instances $Q_1 = [0]P_1$ and $Q_2 = [N_1/4]P_1$, by the cris-cross shading where we have a 'double density' of walks. As the problem instance moves towards $Q_3 = [N_1/2]P_1$ the region of double density of wild walks decreases in size until it disappears. In the case of Q_3 we are now searching exactly the same number of equivalence classes in the tame and wild sets. Then as we move from Q_3 to $Q_4 = [N_1]P_1$ both T and W' are

the same and the analysis follows that of the original Gaudry-Schost algorithm. We define $\mathcal{A}' = T \cap W'$ as the overlap between T and W' with cardinality M'.

Theorem 5.3.2. *Assuming Heuristic 4.1.4, the expected number of group operations to solve the DLP in an interval of size N (without considering the bad steps) using the improved Gaudry-Schost algorithm on equivalence classes is at most*

$$1.47\sqrt{N} + \frac{1}{\theta}$$

group operations.

Proof. Let $Q = [xN]P_1$ for $-1/2 \leq x \leq 1/2$. As in the proof of Theorem 5.2.2 we only need to look at the positive half of the interval of exponents as this is analogous to the number of equivalence classes that we are searching.

1. For problem instances where $0 \leq x \leq 1/4$, there will be a subset of W' with a double density of walks. As explained above in order to model this situation using the Tame-Wild Birthday Paradox we treat these steps as wasted. In practice we can expect to solve the DLP even quicker than specified. The overlap $\mathcal{A}' = W'$ has cardinality $M' = (1/4 + x)N$. So the expected proportion of steps in the $\mathcal{A}'$ is given by

$$\frac{1/4 + x}{1/2}.$$

   So the expected number of group operations before we expect a collision,

scaled over this range of x, is at most

$$2\left(\frac{1}{4}+x\right)\sqrt{\pi\left(\frac{1}{4}+x\right)N} = 2\sqrt{\pi N}\int_{x=0}^{1/4}\left(\frac{1}{4}+x\right)^{3/2}dx \approx 0.207\sqrt{\pi N}. \tag{5.7}$$

2. For problem instances where $1/4 < x \leq 1/2$ there is no subset of W' with a double density of walks. Both T and W' have the same cardinality and the running times are the same as those above. In actual fact the running times that we have integrated in the previous range of x went from the worst case to the best case. So in this range of x the running times are the same but in reverse order i.e. from the best case back down to the worst case. Therefore the expected running time scaled over this range of x is exactly as given in equation (5.7).

We multiply the result from equation (5.7) by 2 and then normalise. This gives the number of group operations before we expect a collision which is at most $1.47\sqrt{N}$. Once a collision has occurred it is only identified when both walks hit a distinguished point and the result follows. □

This is a significant improvement on the original Gaudry-Schost algorithm on equivalence classes. As with all the Gaudry-Schost algorithms bad walks will need to be taken into account when implementing the algorithms but nonetheless, for groups with fast inversion the improved Gaudry-Schost algorithm using equivalence classes is a significant improvement on all previous algorithms. In the next section we give a revised comparison of all the algorithms and some experimental results of the improved Gaudry-Schost algorithm using equivalence classes.

5.4 Comparison and Experimental results

Table 5.1 gives the expected running times (with $\epsilon = 0$, without counting bad steps and omitting the $1/\theta$ terms) of the different algorithms for solving the DLP in an interval of size N. For groups with fast inversion the improved Gaudry-Schost algorithm using equivalence classes is currently the fastest algorithm.

Name of Algorithm	Expected Running time
Van Oorschot & Wiener optimised for the average case	$2\sqrt{N}$
Van Oorschot & Wiener optimised for the worst case	$2.16\sqrt{N}$
Original Gaudry-Schost	$2.08\sqrt{N}$
Improved Gaudry-Schost	$2.05\sqrt{N}$
Pollard 3 kangaroo	$\leq 1.90\sqrt{N}$
Pollard 4 kangaroo	$\leq 1.79\sqrt{N}$
3 set Gaudry-Schost	$\leq 1.85\sqrt{N}$
4 set Gaudry-Schost	$\leq 1.74\sqrt{N}$
Improved 4 set Gaudry-Schost	$\leq 1.72\sqrt{N}$
Original Gaudry-Schost using equivalence classes	$\leq 1.70\sqrt{N}$
Improved Gaudry-Schost using equivalence classes	$\leq 1.47\sqrt{N}$.

Table 5.1: Expected running times of algorithms solving the DLP in an interval of size N (Update 3)

We implemented the Improved Gaudry-Schost algorithm using equivalence classes for solving the DLP in an interval using the software package Magma. The group used was the group of points on the following elliptic curve

$$E : y^2 \equiv x^3 + 40x + 1 \text{ over } \mathbb{F}_p$$

where $p := 3645540875029913$. The group of points has cardinality $\#E(\mathbb{F}_p) = 3645540854261153 > 2^{51}$. We ran a number of experiments on the same interval sizes as those given in Table 4.6. Walks were not permitted to start $2m\sqrt{2/3\pi\theta}$ from the edge of any of the sets. However we found that the chance of a walk starting even close to this new edge was very low. In fact no walks stepped outside of the tame and wild sets. Therefore to achieve a better coverage of the sets we reduced the size of the subsets where walks were not allowed to start to half of the above. The average running times for the different experiments are given in Table 5.2.

	# of Experiments	Improved GS on equivalence classes
Experiment 1 $N \approx 2^{34}$ $m/\theta \approx 2^{14}$	1000	$1.49\sqrt{N}$
Experiment 2 $N \approx 2^{40}$ $m/\theta \approx 2^{19}$	300	$1.47\sqrt{N}$
Experiment 3 $N \approx 2^{48}$ $m/\theta \approx 2^{22}$	50	$1.46\sqrt{N}$

Table 5.2: Average running times of the Improved Gaudry-Schost algorithm using equivalence classes for solving the DLP in an interval of different sizes of N

To detect small cycles we stored the previous 30/35/45 footprints, in Experiment 1,2 and 3 respectively, and compare (as explained in Example 5.1.2); the cost of this is not included in our experimental results. But what is included are the wasted steps from walks which have cycles that were too big to be detected

or walks which were 'trapped' between two cycles. We found that by keeping $\mathcal{T} = 20/\theta$ (the maximum number of steps in a walk), walks which were caught in cycles that could not be detected resulted in too many steps being wasted. This in turn caused the running times to increase. Therefore we instead took $\mathcal{T} = 5/\theta$ and on average the number of steps wasted in this way was approximately $4\%, 7\%, 15\%$ of the total number of steps in Experiment 1, 2 and 3 respectively. Even with this many wasted steps the running times in Table 5.2 are very close to and, in the case of Experiment 3, better than the theoretical result given in Theorem 5.3.2. If we remove the wasted steps from the average running time in Experiment 3 we obtain an average running time of $1.24\sqrt{N}$ group operations. This suggests a more efficient implementation with a different method of detecting cycles may produce better results.

One could use the method of Gallant, Lambert and Vanstone [17, Section 6] for detecting cycles, which would increase the number of group operations but could reduce the number of wasted steps. Alternatively the method of Nivasch [32], which utilises a stack, may lower the running time.

Chapter 6

Multidimensional Discrete Logarithm Problem

In this chapter we consider the multidimensional discrete logarithm problem. The multidimensional DLP is a generalisation of the DLP in an interval. A point to note here is that for all the algorithms in this chapter there is no requirement for the group G to be cyclic.

Definition 6.0.1. Let G be a group of order r. Let $P_1, P_2, \ldots, P_d, Q \in G$ and $N_1, N_2, \ldots, N_d \in \mathbb{N}$ be given. Then the d-dimensional discrete logarithm problem is to find integers $n_i \in [-N_i, N_i]$ for $1 \leq i \leq d$ such that

$$Q = [n_1]P_1 + [n_2]P_2 + \ldots + [n_d]P_d. \tag{6.1}$$

In general we will let N be the size of the entire search space i.e.

$$N = \prod_{i=1}^{d}(2N_i + 1). \tag{6.2}$$

Here we let d be the dimension of the problem. When $d = 1$ we have the problem discussed in Chapters 3, 4 and 5 namely the DLP in an interval. In this chapter it is referred to as the 1-dimensional DLP.

We have seen Shank's BSGS algorithm in Section 2.2 for solving the standard DLP where the interval of possible solutions is only restricted by the order of the group. However we can extend the BSGS algorithm to the multidimensional DLP. The number of group operations required to solve the DLP using this algorithm is $O(\sqrt{N})$ but the main drawback of the algorithm is the storage requirement which is also $O(\sqrt{N})$ group elements. As N gets large there is an obvious motivation to use a low memory algorithm just as in the case of the standard DLP. We have already discussed in depth the algorithms to solve the multidimensional DLP when $d = 1$ so in this chapter we discuss the cases where $d > 1$. Although we will mainly consider the multidimensional DLP for $d \leq 3$ the generic algorithm and the improvements extend to any dimension in theory.

First we motivate the multidimensional DLP in Section 6.1 by discussing various applications where this problem appears. Then we present the standard multidimensional Gaudry-Schost algorithm in Section 6.2. As we encompass many discrete logarithm problems within the umbrella of the multidimensional DLP, we focus on the 2-dimensional problem in Section 6.3.1 then the d-dimensional DLP for $d \geq 3$ in Section 6.4. We present our improved multidimensional Gaudry-Schost algorithm in Sections 6.3.1.2 and 6.4. Our research paper on this algo-

rithm and its analysis [15] was published in the proceedings of the 12th IMA International Conference on Cryptography and Coding in Cirencester, UK.

6.1 Applications of the multidimensional Gaudry-Schost algorithm

Computing the group order of the jacobian of curves can be formulated as a multidimensional DLP. An instance of this is shown in Example 6.1.1.

Example 6.1.1. Let C be a genus 2 curve defined over $\mathbb{F}_q$, a finite field of order q where q is some prime power, then the group order of the jacobian of the curve is given by

$$\#J_C(\mathbb{F}_q) = q^2 + 1 + n_1(1+q) + n_2,$$

where $|n_1| \leq 4\sqrt{q}$ and $|n_2| \leq 6q$ (Weil [49]). If we raise this equation to some generator $P \in J_C(\mathbb{F}_q)$ then we have the following

$$\begin{aligned}[\#J_C(\mathbb{F}_q)]P - [q^2+1]P =& [n_1][1+q]P + [n_2]P \\ \mathcal{O} + Q =& [n_1]P_1 + [n_2]P_2.\end{aligned}$$

We now have a 2-dimensional DLP where n_1 and n_2 are known to be in some bounds.

Gaudry and Schost [19] consider the problem described in Example 6.1.1 on hyperelliptic curves of genus 2 for which the characteristic polynomial of the Frobenius endomorphism is known modulo some integer. In such a case the bounds for n_1 remain the same but the bounds for n_2 become

$$2|n_1|\sqrt{q} - 2q \leq n_2 \leq \frac{n_1^2}{4} + 2q.$$

We will se an improvement on their algorithm for $d > 1$ in Sections 6.3.1.2 and 6.4 but also the extension of the algorithm to all dimensions which we can use for computing the group order of the Jacobian of curves of any genus.

The Gallant, Lambert and Vanstone (GLV) method [18] speeds up elliptic curve arithmetic by rewriting $[n]P$ as $[n_1]P + [n_2]\psi(P)$ for some endomorphism ψ and where $|n_1|, |n_2| \approx \sqrt{n}$. The integers n_1, n_2 are found by solving the closest vector problem in a lattice. An alternative is to choose "smaller" $n_1, n_2 \in \mathbb{Z}$ directly, rather than choosing a random integer n and rewriting it. Solving the DLP for points generated by the GLV method can be phrased as a multidimensional DLP. The methods of this paper imply that n_1 and n_2 cannot be chosen to be too small. See Galbraith and Scott [16] and Galbraith, Lin and Scott [12] for examples of the GLV method with $d > 2$.

Another approach to efficient elliptic curve cryptography is to use Koblitz curves [28]. We rewrite $[n]P$ as

$$\sum_{i=0}^{L} n_i \tau^i(P)$$

where $n_i = \{-1, 0, 1\}$ and $\tau(P) = (x(P)^2, y(P)^2)$ is the 2-power Frobenius map. Since

$$\sum_{i=0}^{L} n_i \tau^i \equiv a + b\tau \quad \text{in} \quad \mathbb{Z}[\tau]/(\tau^2 - t\tau + 2)$$

where $|a| < 3\sqrt{2^L}$ and $|b| < 2\sqrt{2^L}$ as shown by Benits [2], solving the DLP can again be phrased as a multidimensional DLP i.e. $Q = [a]P + [b]\tau(P)$. It follows that L cannot be chosen to be too small. The same ideas can be applied on genus 2 curves over $\mathbb{F}_2$ leading to a 4-dimensional DLP.

6.2 The Multidimensional Gaudry-Schost algorithm

In this section we present the Gaudry-Schost algorithm for the d-dimensional DLP. First we will define the tame and wild sets in this case. From Definition 6.0.1 we know that

$$Q \in \{[a_1]P_1 + [a_2]P_2 + \ldots + [a_d]P_d \mid a_i \in [-N_i, N_i] \text{ for } 1 \leq i \leq d\}.$$

As usual, rather than looking at sets of group elements, we consider sets of d-tuples of exponents. We will also refer to the exponents as the P_i base representation.

Definition 6.2.1. Given the multidimensional DLP as in Definition 6.0.1 then define the tame and wild sets below, respectively,

$$T = \{(a_1, a_2, \ldots, a_d) \mid a_i \in [-N_i, N_i] \text{ for } 1 \leq i \leq d\},$$
$$W = (n_1, n_2, \ldots, n_d) + T.$$

For the purposes of presenting the algorithm we assume there is a function $\mathsf{walk}(x_i, a_{1i}, \ldots, a_{di})$ which computes the next step in the random walk and returns the tuple $(x_{i+1}, a_{1(i+1)}, \ldots, a_{d(i+1)})$. Also we will store the distinguished points in an easily searched structure such as a binary tree. So we will assume searching and storing times are polynomial and therefore will omit these operations from Algorithm 10.

We use the Tame-Wild Birthday Paradox to analyse the running time of the

Algorithm 10 The Gaudry-Schost Algorithm: Server side

INPUT: $d \in \mathbb{N}$, $P_1, P_2, \ldots, P_d, Q \in G$

OUTPUT: Integer tuple $(n_1, n_2, \ldots n_d)$ such that $Q = [n_1]P_1 + [a_2]P_2 + \ldots + [n_d]P_d$

```
Choose the function walk uniformly at random
Let A_T be a binary tree to store distinguished points from tame walks
Let A_W be a binary tree to store distinguished points from wild walks
while multidimensional DLP not solved do
    if received tuple from Tame Processor then
        Construct tuple as z_T := (R, a_1, ..., a_d)
        if (R, b_1, ..., b_d) ∈ A_W for some b_j's (1 ≤ j ≤ d) then
            Send terminate signal to all clients
            return (a_1 − b_1, a_2 − b_2, ..., a_d − b_d)
        else
            Append A_T with z_T
        end if
    else
        Construct tuple as z_W := (R, b_1, ..., b_d)
        if (R, a_1, ..., a_d) ∈ A_T for some a_j's (1 ≤ j ≤ d) then
            Send terminate signal to all clients
            return (a_1 − b_1, a_2 − b_2, ..., a_d − b_d)
        else
            Append A_W with z_W
        end if
    end if
end while
```

Gaudry-Schost algorithm for all dimensions. We can only apply the Tame-Wild Birthday Paradox to the overlap, $\mathcal{A} = T \cap W$, between the tame and wild sets. In the next sections we focus on the 2-dimensional and 3-dimensional DLPs where these sets can be visualised as rectangles and cuboids respectively. In addition we will concentrate on the differences in the Gaudry-Schost algorithm between the 1-dimensional case and when $d > 1$. The two main differences are the pseudorandom walks and the counting of bad steps. More importantly an analogous improvement to that described in Section 4.1.5 for the 1-dimensional case can be applied to the multidimensional case. This is first presented in Section 6.3.1.2 for

Algorithm 11 The Gaudry-Schost Algorithm: Tame Processor

INPUT: $d, N_1, N_2, \ldots N_d \in \mathbb{N}$, $P_1, P_2, \ldots, P_d, Q \in G$, function walk

1: **repeat**
2: Choose random integers $a_j \in [-N_j, N_j]$ for $1 \leq j \leq d$
3: $x := [a_1]P_1 + [a_2]P_2 + \ldots + [a_d]P_d$, Counter:= 0
4: **while** x is not a distinguished point and Counter $\leq 20/\theta$ **do**
5: Counter++
6: $(x, a_1, \ldots, a_d) := \mathsf{walk}(x, a_1, \ldots, a_d)$
7: **end while**
8: Send $(x, a_1, \ldots, a_d)$ to the server.
9: **until** The server sends terminate signal

Algorithm 12 The Gaudry-Schost Algorithm: Wild Processor

INPUT: $d, N_1, N_2, \ldots N_d \in \mathbb{N}$, $P_1, P_2, \ldots, P_d, Q \in G$, function walk

1: **repeat**
2: Choose random integers $a_j \in [-N_j, N_j]$ for $1 \leq j \leq d$
3: $y := Q + [a_1]P_1 + [a_2]P_2 + \ldots + [a_d]P_d$, Counter:= 0
4: **while** y is not a distinguished point and Counter $\leq 20/\theta$ **do**
5: Counter++
6: $(y, a_1, \ldots, a_d) := \mathsf{walk}(y, a_1, \ldots, a_d)$
7: **end while**
8: Send $(y, a_1, \ldots, a_d)$ to the server.
9: **until** The server sends terminate signal

the 2-dimensional problem.

6.3 The 2-dimensional Discrete Logarithm Problem

6.3.1 The Original Gaudry-Schost algorithm

Gaudry and Schost [19] mainly considered the 2-dimensional DLP which was motivated by counting the points on the Jacobian of genus 2 hyperelliptic curves.

To recap given $P_1, P_2, Q \in G$ where the group is $G = E(\mathbb{F}_q)[r]$ as before, the 2-dimensional DLP is to find integers n_1 and n_2 such that $Q = [n_1]P_1 + [n_2]P_2$. We have some extra information that $n_1 \in [-N_1, N_1]$ and $n_2 \in [-N_2, N_2]$ where N_1 and N_2 are less than the order of G. As in equation (6.2) let $N = (2N_1 + 1)(2N_2 + 1)$ but also for ease of notation let $\tilde{N_1} = 2N_1 + 1$ and $\tilde{N_2} = 2N_2 + 1$. The tame and wild sets in this case are,

$$T = \{(a_1, a_2) \mid a_1 \in [-N_1, N_1] \text{ and } a_2 \in [-N_2, N_2]\},$$
$$W = \{(n_1, n_2) + T\}.$$

Unlike the 1-dimensional case we are no longer dealing with an interval of elements but a 'rectangle' of possible solutions where we denote each element by its P_i base representation. As before we define the intersection of the tame and wild sets as $\mathcal{A} = T \cap W$ and its cardinality $M = |\mathcal{A}|$. We use the Tame-Wild Birthday Paradox analysis to obtain the expected running time (expected number of group operations) before we have a TW collision. However there is still an issue of selecting elements of T and W uniformly at random. Unlike the 1-dimensional case there are now many more options for the walk. We will consider a number of different walks in Section 6.3.2 but for now we will use the pseudorandom walk used by Gaudry and Schost which is described below

Pseudorandom Walk - 2D 4. The Gaudry-Schost 2-dimensional pseudorandom walk is as follows: We partition G into distinct sets S_i for $0 \leq i < n_s$ as before. The tame walk starts at $x_1 = [a_1]P_1 + [a_2]P_2$ for some $a_1 \in [-N_1, N_1]$ and

$a_2 \in [-N_2, N_2]$ and continues as follows

$$x_{i+1} = f(x_i) = x_i + [1]P_1 + [z_j]P_2$$

where the steps z_j for $0 \leq j \leq n_s$ are uniformly distributed at random in the interval $[-2m_2, 2m_2]$ where m_2 is the mean step size in the P_2 component. The wild walks start at $y_1 = Q + [a_1]P_1 + [a_2]P_2$ with a_1 and a_2 as above and continue as above.

6.3.1.1 Running time analysis

Gaudry and Schost use a Tame-Wild Birthday Paradox analysis on the overlap $\mathcal{A}$ to obtain an expected running time. As before we will need to consider the wasted or 'bad' steps of the algorithm. This is looked at in Section 6.3.3 after we have observed any possible improvements to their algorithm in the 2-dimensional case. If N is sufficiently large then we can make the following assumption,

Heuristic 6.3.1. If N is sufficiently large then the pseudorandom walk described in Pseudorandom walk 4 is sufficiently random that the Tame-Wild Birthday Paradox analysis applies to the set $\mathcal{A}$ in the 2-dimensional discrete logarithm problem.

We will come back to Heuristic 6.3.1 in Section 6.3.4. In that section we will also study what sorts of pseudorandom walks can arise when working in 2 dimensions. As in the 1-dimensional case we take the number of processors N_P to be 1. Unlike the 1-dimensional algorithm the cardinality of the overlap can have the following range $\frac{N}{4} \leq M \leq N$.

Gaudry and Schost [19, Section 4.1] outline the following result which we also prove.

Theorem 6.3.2. *Assuming Heuristic 6.3.1 the expected number of group operations to solve the 2-dimensional DLP of size N using the original Gaudry-Schost algorithm (without considering the bad steps) is*

$$2.43\sqrt{N} + \frac{1}{\theta}. \tag{6.3}$$

Proof. To find the average expected running time we have to average over all possible n_1 and n_2. Without loss of generality let us consider problem instances in one 'corner' of the search space so let $Q = [-N_1 + x\tilde{N}_1]P_1 + [-N_2 + y\tilde{N}_2]P_2$ for $x, y \in [0, 1/2]$. The size of the overlap $\mathcal{A}$ is then

$$\left(\frac{1}{2} + x\right)\tilde{N}_1 \cdot \left(\frac{1}{2} + y\right)\tilde{N}_2 = \left(\frac{1}{2} + x\right)\left(\frac{1}{2} + y\right)N.$$

So the expected proportion of tame or wild walks in $\mathcal{A}$ is

$$\left(\left(\frac{1}{2} + x\right)\left(\frac{1}{2} + y\right)\right)^{-1}.$$

So the expected number of group operations before we have a TW-collision is

given by

$$
\begin{aligned}
&4\int_{x=0}^{1/2}\int_{y=0}^{1/2}\frac{1}{\left(\frac{1}{2}+x\right)\left(\frac{1}{2}+y\right)}\sqrt{\pi\left(\frac{1}{2}+x\right)\left(\frac{1}{2}+y\right)N}\,dy\,dx\\
&=4\sqrt{\pi N}\int_{x=0}^{1/2}\int_{y=0}^{1/2}\left(\left(\frac{1}{2}+x\right)\left(\frac{1}{2}+y\right)\right)^{-1/2}dy\,dx\\
&=4\sqrt{\pi N}\left(\int_{x=0}^{1/2}\left(\frac{1}{2}+x\right)^{-1/2}dx\right)^2\\
&=16\sqrt{\pi N}\left(1-(1/2)^{1/2}\right)^2\\
&\approx 2.43284\sqrt{N}.
\end{aligned}
$$

Then the walk will need to continue to the distinguished point and we have the result. □

6.3.1.2 Improving the Gaudry-Schost Algorithm in the 2-dimensional case

As in the 1-dimensional case in order that we have a constant running time we would like to make the overlap between the tame and wild sets constant for all problem instances. Using the analogous strategy from the 1-dimensional case we will search in a 'rectangle' of size k^2N, where $0 < k < 1$, centred in the middle of the tame set and we will search the same number of elements in the wild set but split up into four disjoint sets at the 'corners' of W. The 'New Tame' set is given below,

$$T' = \{(a_1, a_2) \mid a_1 \in [-kN_1, kN_1] \text{ and } a_2 \in [-kN_2, kN_2]\}.$$

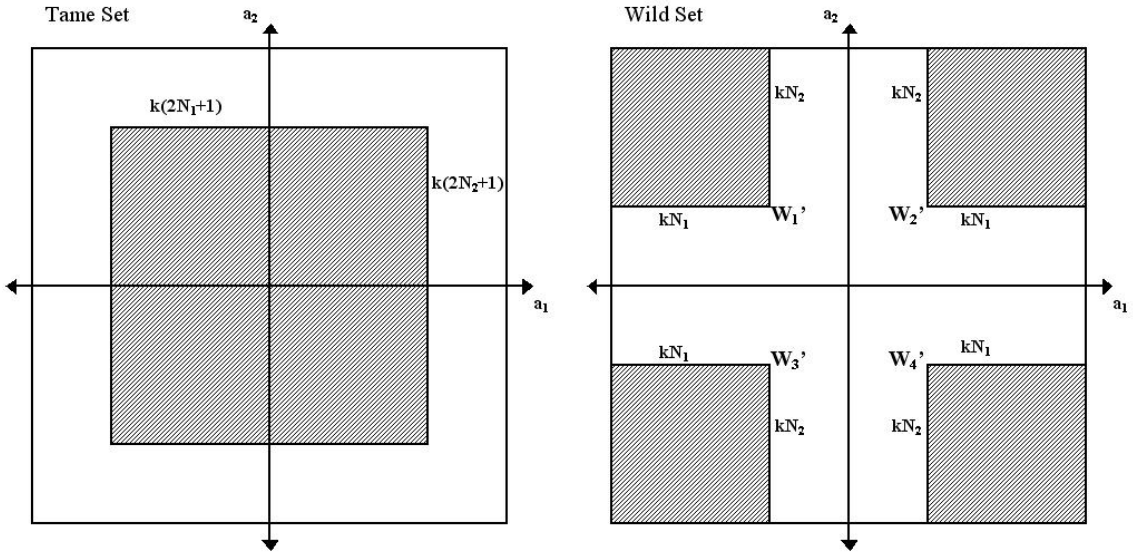

Figure 6.1: Searching k^2N of the Tame and Wild Sets

The 'New Wild' set can be written as the union of four disjoint sets, $W' = W_1' \cup W_2' \cup W_3' \cup W_4'$, which are given below,

$$W_1' = \{(n_1, n_2) + (a_1, a_2) \mid a_1 \in [-N_1, (k-1)N_1] \text{ and } a_2 \in [(1-k)N_2, N_2]\}$$
$$W_2' = \{(n_1, n_2) + (a_1, a_2) \mid a_1 \in [(1-k)N_1, N_1] \text{ and } a_2 \in [(1-k)N_2, N_2]\}$$
$$W_3' = \{(n_1, n_2) + (a_1, a_2) \mid a_1 \in [-N_1, (k-1)N_1] \text{ and } a_2 \in [-N_2, (k-1)N_2]\}$$
$$W_4' = \{(n_1, n_2) + (a_1, a_2) \mid a_1 \in [(1-k)N_1, N_1] \text{ and } a_2 \in [-N_2, (k-1)N_2]\}.$$

Figure 6.1 shows the new tame and wild sets diagrammatically. In the previous best case Q lay in the middle of the 'rectangle', so the new overlap $\mathcal{A}' = T' \cup W'$ in this instance has cardinality

$$M' = 4\left(\frac{\tilde{N}_1(2k-1)}{2}\right)\left(\frac{\tilde{N}_2(2k-1)}{2}\right) = (2k-1)^2\tilde{N}_1\tilde{N}_2 = (2k-1)^2N.$$

In the previous worst case Q lay in a corner of the 'rectangle', so the new overlap has cardinality,

$$M' = \frac{k^2 N}{4}.$$

Equating these different values of M gives the following quadratic

$$15k^2 - 16k + 4 = 0,$$

which has solutions $k = 2/5$ or $2/3$. We can eliminate the first as a solution as the overlap would be empty in the extreme case. So we take $k = 2/3$.

Lemma 6.3.3. *Using the notation above when $k = 2/3$ the cardinality of the overlap between the new tame and wild sets T' and W' respectively is*

$$|\mathcal{A}'| = M' = \frac{N}{9}$$

for all problem instances Q.

Proof. Without loss of generality let us deal with the problem instances where $n_1, n_2 \geq 0$. We know that when the problem instance is $Q_1 = [0]P_1 + [0]P_2$ or $Q_2 = [N_1]P_1 + [N_2]P_2$ the overlap is $N/9$. Let us start at Q_1 and look at the problem instance as it moves to Q_2. As the problem instance moves in the P_1 component $T' \cap W_2'$ and $T' \cap W_3'$ increases at the same amount as $T' \cap W_1'$ and $T' \cap W_4'$ decreases. An analogous event occurs when we move the problem instance in the P_2 component (vertical movement in Figure 6.1). Therefore we can see that M' will remain constant for all problem instances hence proving the result. □

Theorem 6.3.4. *Assuming Heuristic 6.3.1 the expected number of group operations to solve the 2-dimensional DLP of size N using the improved Gaudry-Schost algorithm when $k = 2/3$ (without considering the bad steps) is*

$$\frac{4\sqrt{\pi N}}{3} + \frac{1}{\theta} \approx 2.36\sqrt{N} + \frac{1}{\theta}. \tag{6.4}$$

Proof. From Lemma 6.3.3 we know that the overlap for all problem instances is $N/9$. On average one quarter of steps will be in the overlap $\mathcal{A}'$ just as in the previous worst case. However the new overlap is smaller than in the original algorithm and so the expected number of steps before we have a TW collision is

$$4\sqrt{\frac{\pi N}{9}} = \frac{4\sqrt{\pi N}}{3}.$$

Then the walk will need to continue to the distinguished point and we have the result. □

We can see that this is an improvement on the original Gaudry-Schost algorithm (Theorem 6.3.2) with the added bonus that this will be the running time for all problem instances. Gaudry and Schost did not analyse any form of bound for the expected number of bad steps in their paper [19]. We will cover that in the Section 6.3.3. However, bounding bad steps depends on the type of walk we are using so first we will consider a number of different walks.

6.3.2 Pseudorandom walks in the 2-dimensional case

Now we consider a smattering of different pseudorandom walks which can be implemented in the 2-dimensional case and then in Section 6.3.3 we will count the bad steps for the different types of walks.

The tame walks start $x_1 = [a_1]P_1 + [a_2]P_2$ for some $a_1 \in [-N_1, N_1]$ and $a_2 \in [-N_2, N_2]$ and continue as per the pseudorandom walks below. The wild walks are very similar except that they start $y_1 = Q + [a_1]P_1 + [a_2]P_2$ with a_1 and a_2 as above and continue as per the pseudorandom walks below. Note that for walks of type '1-forward and to the right' we assume $N_2 \geq N_1$ and for the other types of walks we assume the opposite, i.e. that $N_1 \geq N_2$.

Definition 6.3.5. Let m_1 and m_2 be the mean step size in the P_1 and P_2 components respectively. Where a walk has both positive and negative steps in a particular component then the associated m_i will be the mean absolute step size which is calculated by taking the mean of the absolute values of the steps sizes.

Pseudorandom Walk - 2D Type (1-forward and to the right)**.** The number of partitions of G is n_s and $0 \leq S(x_i) < n_s$.

$$x_{i+1} = f(x_i) = x_i + [1]P_1 + [2^{S(x_i)}]P_2.$$

Pseudorandom Walk - 2D Type (1-forward and side-to-side)**.** The number of partitions of G is n_s, $0 \leq S(x_i) < n_s$ and b is some base i.e. $b = 2$.

$$x_{i+1} = f(x_i) = \begin{cases} x_i + [1]P_1 + [b^{S(x_i)/2}]P_2 & \text{if } S(x_i) \equiv 0 \pmod 2 \\ x_i + [1]P_1 + [-b^{(S(x_i)-1)/2}]P_2 & \text{if } S(x_i) \equiv 1 \pmod 2. \end{cases}$$

Pseudorandom Walk - 2D Type (1-forward and side-to-side uniformly)**.** The number of partitions of G is n_s

$$x_{i+1} = f(x_i) = x_i + [1]P_1 + [z_j]P_2$$

where the steps z_j for $0 \leq j \leq n_s$ are uniformly distributed at random in the interval $[-2m_2, 2m_2]$. *N.B. This is the pseudorandom walk used by Gaudry and Schost [19]*

In Section 6.3.4 we will run experiments on the different pseudorandom walks above to compare which is closest to selecting elements uniformly at random. However we need to consider the different types of walks before we look at the bad steps as the bounds on the expected number of bad steps depends on the type of pseudorandom walk. The three types above are: -

- '1-forward and to the right': This pseudorandom walk has no 'side-to-side' aspect and in that sense this walk works very much like the walk in the 1-dimensional case. We will bound the number of bad steps of this walk in Section 6.3.3.1.

- '1-forward and side-to-side': which are pseudorandom walks that step 'side-to-side' in the P_2 component in powers of some base b. We will consider this walk and variations in Section 6.3.4. To stop the walks visiting an element with the same exponents we always move in a positive direction in a particular component namely P_1 where the mean step size is 1 for all the walks. We will bound the number of bad steps of this walk in Section 6.3.3.2.

- '1-forward and side-to-side uniformly': This pseudorandom walk is very similar to the previous walk except the steps are not powers of a base but are uniformly distributed in $[-2m_2, 2m_2]$. In every other way these walks are the same as the '1-forward and side-to-side' pseudorandom walks. We will bound the number of bad steps of this walk in Section 6.3.3.2 as well.

6.3.3 Counting bad steps in the 2-dimensional case

In this section we give bounds on the number of wasted or 'bad' steps. As a reminder the first type of bad step comes from a walk which never reaches a distinguished point and the second type comes from walks which step outside of the tame and wild sets and therefore cannot be added to the Tame-Wild Birthday Paradox. Using Lemma 4.1.9 we can again bound the bad steps of type 1 as 5×10^{-8} of the total number of steps. This will be true for all dimensions.

Definition 6.3.6. Let B_1 and B_2 denote the number of bad steps of type 2 in the P_1 and P_2 components respectively.

6.3.3.1 '1-forward and to the right' walks

For pseudorandom walks of type '1-forward and to the right',

$$m_2 = \frac{2^{n_s} - 1}{n_s},$$

The maximum step size in the P_2 component is $\max_2 = 2^{n_s - 1}$. In the case of the tame walks, by Lemma 4.1.3, walks are not permitted to start $c_B = \frac{1}{\theta}$ from the right hand edge of the interval of a_1 and $d_B = \frac{m_2}{\theta}$ from the right hand edge of

the interval of a_2 as shown in Figure 6.2. So the subsets of T', W_1', W_2', W_3' and W_4' where walks are permitted to start are given by the following respectively,

$$T' = \{(a_1, a_2) \mid a_1 \in [-2N_1/3, 2N_1/3 - 1/\theta] \text{ and } a_2 \in [-2N_2/3, 2N_2/3 - m_2/\theta]\},$$

$$W_1' = \{(n_1, n_2) + (a_1, a_2) \mid a_1 \in [-N_1, -N_1/3 - 1/\theta] \text{ and } a_2 \in [N_2/3, N_2 - m_2/\theta]\},$$

$$W_2' = \{(n_1, n_2) + (a_1, a_2) \mid a_1 \in [(N_1/3, N_1 - 1/\theta] \text{ and } a_2 \in [N_2/3, N_2 - m_2/\theta]\},$$

$$W_3' = \{(n_1, n_2) + (a_1, a_2) \mid a_1 \in [-N_1, -N_1/3 - 1/\theta] \text{ and } a_2 \in [-N_2, -N_2/3 - m_2/\theta]\},$$

$$W_4' = \{(n_1, n_2) + (a_1, a_2) \mid a_1 \in [N_1/3, N_1 - 1/\theta] \text{ and } a_2 \in [-N_2, -N_2/3 - m_2/\theta]\}.$$

In Figure 6.2 we can see these areas where walks are not permitted to start on

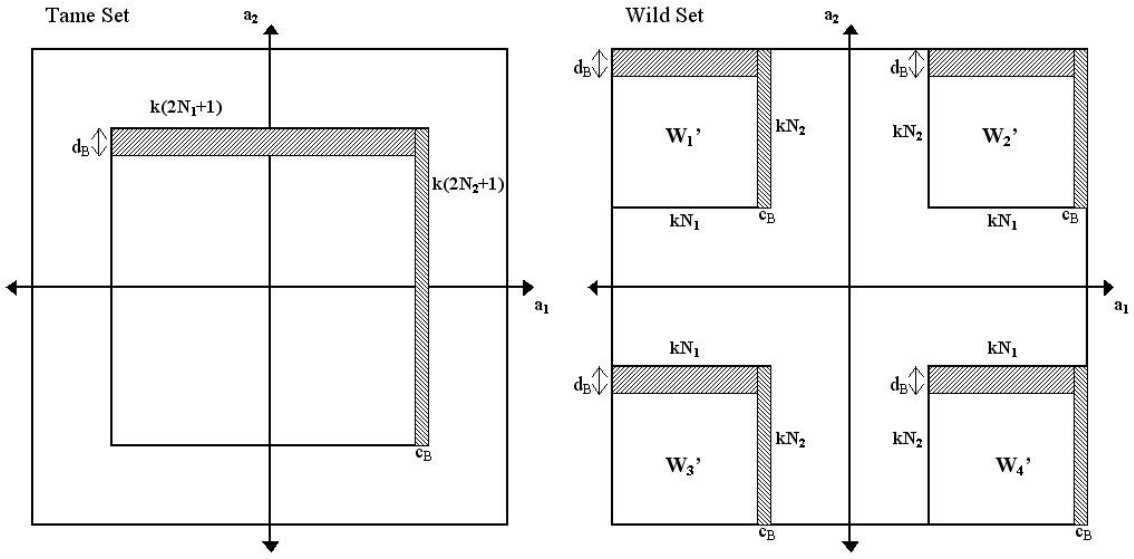

Figure 6.2: Shaded area represents the area where walks cannot start for pseudorandom walks of type '1-forward and to the right'

the tame set and also how they correspond on the wild set. As W' is the union of four disjoint sets the area where wild walks are not permitted to start is not the same as in T'. In fact just as in the 1-dimensional case our bound on the number

of bad steps in the W' will be just over twice that of T. So we will concentrate on the wild set and adjust at the end by bounding the tame set component as half that of the wild set.

Lemma 6.3.7. *Given that the algorithm has made K steps, then we can bound the P_1 component of the number of bad steps of type 2 in the 2-dimensional '1-forward and to the right' case as,*

$$B_1 < \frac{4.365K}{\tilde{N_1}\theta/3 - 1}. \tag{6.5}$$

Proof. We will focus on the wild walks as we have seen that the number of bad steps of type 2 from wild walks are just over twice as those in the tame walks. Without loss of generality let us consider the set W_2'. Let X denote the set of all elements of W_2' such that if a pseudorandom walk starts at those elements there is a nonzero probability that the walk will step outside of the search space in its P_1 component. Unlike the 1-dimensional case, we cannot be as crude as saying that every walk that has a possibility of stepping outside W_2' will do so with probability 1. We have to consider the set X in distinct bands of width $\frac{1}{\theta}$. Then in each band this probability will be different i.e. it will be smaller as we go further into the search space. So we can write

$$X = \bigcup_{i=1}^{19} X_i$$

where

$$X_i = \left\{ (a_1, a_2) \;\middle|\; a_1 \in \left[N_1 - \frac{i+1}{\theta}, N_1 - \frac{i}{\theta} \right] \text{ and } a_2 \in \left[\frac{N_2}{3}, N_2 - \frac{m_2}{\theta} \right] \right\}.$$

Then an upper bound on B_1 in band X_i can be given by the following calculation: The probability that a wild walk starts in X_i given that it is already starting in $W_2' \times$ a bound on the probability that a walk will step outside $W_2' \times$ the expected number of wild walks in $W_2' \times \mathcal{T}$. So the probability that a walk starts in X_i is

$$\frac{\frac{1}{\theta}\left(\frac{\tilde{N}_2}{3} - \frac{m_2}{\theta}\right)}{\left(\frac{\tilde{N}_1}{3} - \frac{1}{\theta}\right)\left(\frac{\tilde{N}_2}{3} - \frac{m_2}{\theta}\right)} = \frac{1}{\tilde{N}_1\theta/3 - 1}.$$

Therefore we have an upper bound for the wild walk part in W_2' of the number of bad steps of type 2 in the P_1 component which is

$$\begin{aligned}\left[\sum_{i=1}^{19} \frac{1}{\tilde{N}_1\theta/3 - 1} \cdot (1-\theta)^{\frac{i}{\theta}}\right] \cdot \frac{K\theta}{16} \cdot \frac{20}{\theta} \leq & \frac{5}{4} K \sum_{i=1}^{19} \frac{e^{-i}}{\tilde{N}_1\theta/3 - 1} \\ \leq & \frac{0.727K}{\tilde{N}_1\theta/3 - 1}.\end{aligned}$$

Multiplying the total by 4 to take into account all disjoint parts of W' and then multiplying by 3/2 to take into account both the tame and wild walks we have the result. □

Lemma 6.3.8. *Given that the algorithm has made K steps, then we can bound the P_2 component of the number of bad steps of type 2 in the 2-dimensional '1-forward and to the right' case as,*

$$B_2 < \frac{11.86 \max_2 K}{\tilde{N}_2\theta/3 - m_2}. \tag{6.6}$$

Proof. We will focus on the wild walks as we have seen that the number of bad steps of type 2 from wild walks are just over twice as those in the tame walks. Without loss of generality let us consider the set W_2'. Let X denote the set of all

elements of W_2' such that if a pseudorandom walk starts at those elements there is a nonzero probability that the walk will step outside of the search space in its P_2 component. As in the case of B_1, we have to consider X in distinct bands of width $\frac{\max_2}{\theta}$. Then in each band this probability will be different i.e. it will be smaller as we go further into the search space. However the first band will not have the same width as the others. As the mean step size m_2 is different from the maximum step size $\max_2$, the first band will have width $\frac{\max_2 - m_2}{\theta}$ whilst the others will have width $\frac{\max_2}{\theta}$. So we can write

$$X = \bigcup_{i=0}^{19} X_i,$$

where

$$X_0 = \left\{ (a_1, a_2) \;\middle|\; a_1 \in \left[\frac{N_1}{3}, N_1 - \frac{1}{\theta}\right] \text{ and } a_2 \in \left[N_2 - \frac{\max_2}{\theta}, N_2 - \frac{m_2}{\theta}\right] \right\}$$
$$X_i = \left\{ (a_1, a_2) \;\middle|\; a_1 \in \left[\frac{N_1}{3}, N_1 - \frac{1}{\theta}\right] \text{ and } a_2 \in \left[N_2 - \frac{(i+1)\max_2}{\theta}, N_2 - \frac{i\max_2}{\theta}\right] \right\}$$

for $1 \leq i \leq 19$. Therefore we have an upper bound for the wild walk part of W_2' of the number of bad steps of type 2 in the P_2 component which is

$$\begin{aligned}
&\left[\left(\frac{\max_2 - m_2}{\tilde{N}_2\theta/3 - m_2}\,(1-\theta)^{\frac{\max_2 - m_2}{\max_2\,\theta}}\right) + \sum_{i=1}^{19} \frac{\max_2}{\tilde{N}_2\theta/3 - m_2} \cdot (1-\theta)^{\frac{i}{\theta}}\right] \cdot \frac{K\theta}{16} \cdot \frac{20}{\theta} \\
&\leq \frac{5}{4} K \left[\frac{(\max_2 - m_2)e^{-1+\frac{m_2}{\max_2}}}{\tilde{N}_2\theta/3 - m_2} + \sum_{i=1}^{19} \frac{\max_2 e^{-i}}{\tilde{N}_2\theta/3 - m_2}\right] \\
&\leq \frac{1.977 \max_2 K}{\tilde{N}_2\theta/3 - m_2}.
\end{aligned}$$

Multiplying the total by 4 to take into account all disjoint parts of W' and then

multiplying by 3/2 to take into account both the tame and wild walks we have the result. □

We can see from Lemma 6.3.7 and 6.3.8 that the bounds on B_1 and B_2 depend upon $\tilde{N_1}$ and $\tilde{N_2}$ (respectively), θ and n_s or another way to look at it; if we want to limit these bounds to a certain proportion of the total number of steps, i.e. 1%, then for specific $\tilde{N_1}$ and $\tilde{N_2}$ there may be limits to how large θ and n_s can be. Table 6.1 gives the minimum bounds for $\tilde{N_1}$ if we want to keep B_1 as 0.5% of the total number of steps for different values of θ. In this case the value of n_s is redundant as m_1 for all n_s is 1. Table 6.2 gives the minimum bounds for $\tilde{N_2}$ if we want to keep B_2 as 0.5% of the total number of steps for different values of θ and n_s.

$\theta = 2^{-8}$	$\theta = 2^{-16}$	$\theta = 2^{-32}$
2^{20}	2^{28}	2^{44}

Table 6.1: Minimum values of $\tilde{N_1}$ to have $B_1 < 0.5\%$ of the total number of steps for different θ

	$\theta = 2^{-8}$	$\theta = 2^{-16}$	$\theta = 2^{-32}$
$n_s = 8$	2^{28}	2^{36}	2^{52}
$n_s = 16$	2^{36}	2^{44}	2^{60}
$n_s = 20$	2^{40}	2^{48}	2^{64}
$n_s = 32$	2^{52}	2^{60}	2^{76}

Table 6.2: Minimum values of $\tilde{N_2}$ to have $B_2 < 0.5\%$ of the total number of steps for different θ and n_s for pseudorandom walks of type '1-forward and to the right'

6.3.3.2 '1-forward and side-to-side' walks

With the '1-forward and side-to-side' and the '1-forward and side-to-side uniformly' walks we have to look at the mean steps sizes differently. In the P_2 component the mean step size of the side-to-side pseudorandom walks is 0 so we now let m_2 be the mean absolute step size which is calculated by taking the mean of the absolute values of the step sizes. Our pseudorandom walks, as described in

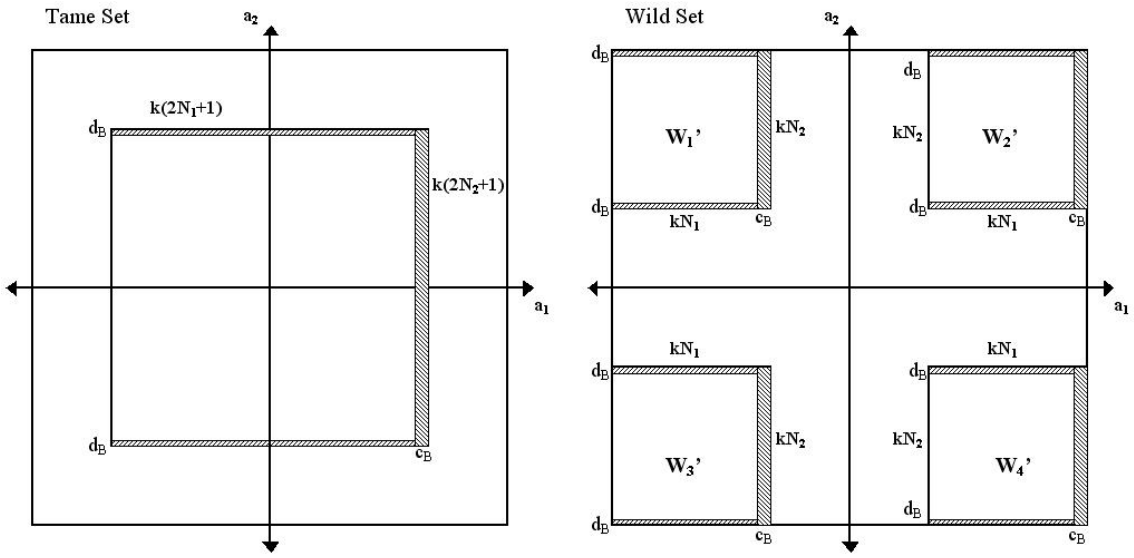

Figure 6.3: Shaded area represents the area where walks cannot start for pseudorandom walks of type '1-forward and side-to-side'

Section 6.3.2, have the same mean step size in the P_1 component as the '1-forward and to the right' walks so the boundary where walks are not permitted to start in the P_1 component is of size $c_B = \frac{1}{\theta}$, as before. Therefore an upper bound on the number of bad steps of type 2 in the P_1 component, B_1, also remains the same and is given in Lemma 6.3.7. However in the P_2 component we have 'side-to-side' walks and therefore calculate our bound B_2 differently than in the '1-forward and to the right' walks. We again use the result of Cofman, Flajolet, Flatto and Hofri

[7] (as described in Lemma 5.1.1) to give the expected maximum distance (or expected maximum excursion) that our walks will go in the P_2 component from its starting position. This will become d_B, as shown in Figure 6.3, which will be on both extreme ends of the interval of the P_2 component. The pseudorandom walks of type '1-forward and side-to-side' have step sizes in the P_2 component which are powers of a base b and therefore are not uniformly spread out between the smallest and largest steps. However when defining d_B we will treat these walks in the same way as those of type '1-forward and side-to-side uniformly'. Therefore we have the following Heuristic.

Heuristic 6.3.9. As the number of partitions of G increases a 'side to side' pseudorandom walk, which takes steps uniformly at random in the interval $[-2m_2, 2m_2]$, will have expected maximum excursion

$$2m_2\sqrt{\frac{2n}{3\pi}}$$

after n steps, as given in Lemma 5.1.1.

Heuristic 6.3.9 is a realistic assumption and in Section 6.3.4 we will compare our pseudorandom walks to see if walks which are powers of a base b perform any differently in practice to those which takes steps uniformly at random in the interval $[-2m_2, 2m_2]$.

Lemma 6.3.10. *Assuming Heuristic 6.3.9 and that our '1-forward and side-to-side' pseudorandom walk takes steps uniformly at random in the interval* $[-2m_2, 2m_2]$, *then in the* P_2 *component walks should not start less then* d_B *from the righthand*

edge of the interval where

$$d_B = m_2\sqrt{\frac{8}{3\pi\theta}}.$$

Proof. Assuming Heuristic 6.3.9, from Theorem 3.2.1 the expected walk length is $\frac{1}{\theta}$ and our mean absolute step size is $m_2 = 2m_2 \times \frac{1}{2}$. Therefore the result follows directly from Lemma 5.1.1. □

So the subsets of T', W_1', W_2', W_3' and W_4' where walks are permitted to start are given by the following respectively,

$$\begin{aligned}
T' &= \left\{ (a_1, a_2) \;\middle|\; a_1 \in \left[-\frac{2N_1}{3}, \frac{2N_1}{3} - 1/\theta\right] \text{ and } a_2 \in \left[-\frac{2N_2}{3} + d_B, \frac{2N_2}{3} - d_B\right] \right\}, \\
W_1' &= \left\{ (n_1, n_2) + (a_1, a_2) \;\middle|\; a_1 \in \left[-N_1, -\frac{N_1}{3} - \frac{1}{\theta}\right] \text{ and } a_2 \in \left[\frac{N_2}{3} + d_B, N_2 - d_B\right] \right\}, \\
W_2' &= \left\{ (n_1, n_2) + (a_1, a_2) \;\middle|\; a_1 \in \left[\frac{N_1}{3}, N_1 - \frac{1}{\theta}\right] \text{ and } a_2 \in \left[\frac{N_2}{3} + d_B, N_2 - d_B\right] \right\}, \\
W_3' &= \left\{ (n_1, n_2) + (a_1, a_2) \;\middle|\; a_1 \in \left[-N_1, -\frac{N_1}{3} - \frac{1}{\theta}\right] \text{ and } a_2 \in \left[-N_2 + d_B, -\frac{N_2}{3} - d_B\right] \right\}, \\
W_4' &= \left\{ (n_1, n_2) + (a_1, a_2) \;\middle|\; a_1 \in \left[\frac{N_1}{3}, N_1 - \frac{1}{\theta}\right] \text{ and } a_2 \in \left[-N_2 + d_B, -\frac{N_2}{3} - d_B\right] \right\}.
\end{aligned}$$

An important point to note here is that when we consider the side-to-side walks in practice, it is possible that they can step out and then back into the search area. Such walks are useful in practice but cannot be counted towards the Tame-Wild Birthday Paradox analysis.

Lemma 6.3.11. *Given that the algorithm has made K steps, then we can bound the P_2 component of the number of bad steps of type 2 in the 2-dimensional '1-*

forward and side-to-side' case as,

$$B_2 < \frac{12.48 \max_2 K}{\tilde{N}_2 \theta/3 - 2\theta d_B} \tag{6.7}$$

where $\max_2$ *is the largest positive step size.*

Proof. In the same way as the '1-forward and to the right' walks the number of bad steps of type 2 from the wild walks are just over twice as those in the tame walks so we will focus on the wild walks and adjust our calculation at the end to include all walks. Without loss of generality let us consider the set W_2'. Let X denote the set of all elements of W_2' such that if a pseudorandom walk starts at those elements there is a nonzero probability that the walk will step outside of the search space in its P_2 component. Then $X = X_L \cup X_R$ where

$$X_L = \left\{ (a_1, a_2) \;\middle|\; a_1 \in \left[\frac{N_1}{3}, N_1 - \frac{1}{\theta} \right] \text{ and } a_2 \in \left[\frac{N_2}{3} + d_B, \frac{N_2}{3} + \frac{20 \max_2}{\theta} \right] \right\}$$
$$X_R = \left\{ (a_1, a_2) \;\middle|\; a_1 \in \left[\frac{N_1}{3}, N_1 - \frac{1}{\theta} \right] \text{ and } a_2 \in \left[N_2 - \frac{20 \max_2}{\theta}, N_2 - d_B \right] \right\}$$

Just as in the proof of Lemma 6.3.8 we have to consider each of these sets in distinct bands of width $\frac{\max_2}{\theta}$. Then in each band the probability of stepping outside of the search space will be smaller as we go further into the search space. However, as before, the first band will have a different width, being $\frac{\max_2}{\theta} - d_B$. Let us consider X_R. We can write

$$X_R = \bigcup_{i=0}^{19} X_{Ri},$$

where

$$X_{R0} = \left\{ (a_1, a_2) \;\middle|\; a_1 \in \left[\frac{N_1}{3}, N_1 - \frac{1}{\theta} \right] \text{ and } a_2 \in \left[N_2 - \frac{\max_2}{\theta}, N_2 - d_B \right] \right\}$$
$$X_{Ri} = \left\{ (a_1, a_2) \;\middle|\; a_1 \in \left[\frac{N_1}{3}, N_1 - \frac{1}{\theta} \right] \text{ and } a_2 \in \left[N_2 - \frac{(i+1)\max_2}{\theta}, N_2 - \frac{i \max_2}{\theta} \right] \right\}$$

for $1 \leq i \leq 19$. Then an upper bound on B_2 in band X_{Ri} can be given by the following calculation: The probability that a wild walk starts in X_{Ri} given that it is in $W_2' \times$ a bound on the probability that a walk will step outside W_2' in its P_2 component $\times$ the expected number of wild walks in $W_2' \times \mathcal{T}$. In the first band, X_{R0}, a bound on the probability that a walk will step outside W_2' is 0.25 as we expect half of the walks to walk beyond the expected maximum excursion and we can expect half of those to be on the right of the starting point (outside the search space) as opposed to the left. So the probability that a walk starts in X_{R0} is

$$\frac{\frac{\max_2}{\theta} - d_B}{\tilde{N}_2/3 - 2d_B} = \frac{\max_2 - \theta d_B}{\tilde{N}_2\theta/3 - 2\theta d_B}$$

Therefore we have an upper bound for the right hand wild walk part of W_2' of the number of bad steps of type 2 in the P_2 component which is

$$\begin{aligned}
&\left[\left(\frac{\max_2 - \theta d_B}{\tilde{N}_2\theta/3 - 2\theta d_B} \cdot 0.25 \right) + \sum_{i=1}^{19} \frac{\max_2}{\tilde{N}_2\theta/3 - 2\theta d_B} \cdot (1-\theta)^{\frac{i}{\theta}} \right] \cdot \frac{K\theta}{16} \cdot \frac{20}{\theta} \\
&\leq \frac{5}{4} K \left[0.25 \frac{\max_2 - \theta d_B}{\tilde{N}_2\theta/3 - 2\theta d_B} + \sum_{i=1}^{19} \frac{\max_2 e^{-i}}{\tilde{N}_2\theta/3 - 2\theta d_B} \right] \\
&\leq \frac{1.04 \max_2 K}{\tilde{N}_2\theta/3 - 2\theta d_B}.
\end{aligned}$$

Doubling this gives us the W_2' component of B_2. Then multiplying by 4 to take into account all disjoint parts of W' and then multiplying by 3/2 to take into

account both the tame and wild walks we have the result. □

Table 6.1 again gives the minimum bounds for $\tilde{N}_1$ for the different values of θ and Table 6.3 below gives us the minimum bounds for $\tilde{N}_2$ to have B_2 as 0.5% of the total number of steps. Due to Lemma 5.1.1 the bound B_2 is dependent on the mean absolute step size, m_2, which is unrelated to n_s in a general pseudorandom walk. In Table 6.3 the mean absolute step sizes are similar to the mean step size in Table 6.2.

	$\theta = 2^{-8}$	$\theta = 2^{-16}$	$\theta = 2^{-32}$
$m_2 = 2^5$	2^{27}	2^{35}	2^{51}
$m_2 = 2^{12}$	2^{34}	2^{42}	2^{58}
$m_2 = 2^{15}$	2^{37}	2^{45}	2^{61}
$m_2 = 2^{27}$	2^{49}	2^{57}	2^{73}

Table 6.3: Minimum values of $\tilde{N}_2$ to have $B_2 < 0.5\%$ of the total number of steps for different θ and m_2 for pseudorandom walks of type '1-forward and side-to-side'

6.3.4 Experiments in the 2-dimensional case

The experiments that we ran in the 2-dimensional case have two principle aims. The first is to check Heuristic 6.3.1, namely that our pseudorandom walks behave close enough to selecting elements at random from the tame and wild sets. The second is to compare walks with steps that are powers of a base b (as used by Pollard [34, 35]) to those walks whose steps are integers chosen at random once at the start of the experiment (as used by Gaudry-Schost). For each pseudorandom walk 1000 experiments were run on a search space approximately 2^{43} in size. We simulated a group and recorded the number of group operations that were

made before a collision between 2 walks was found. To test our assumptions we compared the expected number of group operations given by the Birthday Paradox and the actual number of group operations. This gave us the following ratio

$$\frac{\text{The Actual Number of Group operations}}{\text{The Expected number of Group Operations}}.$$

The pseudorandom walks that we simulated are as follows

Pseudorandom Walk - 2D 5. The number of partitions of G are $n_s = 16$. $m_1 = 1$ and $m_2 = 31.875$.

$$x_{i+1} = f(x_i) = x_i + [1]P_1 + [2^{S(x_i)}]P_2.$$

Pseudorandom Walk - 2D 6. The number of partitions of G are $n_s = 8$. $m_1 = 1$ and $m_2 = 3.75$.

$$x_{i+1} = f(x_i) = \begin{cases} x_i + [1]P_1 + [2^{S(x_i)/2}]P_2 & \text{if } S(x_i) \equiv 0 \pmod 2 \\ x_i + [1]P_1 + [-2^{(S(x_i)-1)/2}]P_2 & \text{if } S(x_i) \equiv 1 \pmod 2. \end{cases}$$

Pseudorandom Walk - 2D 7. The number of partitions of G are $n_s = 16$. $m_1 = 1$ and $m_2 = 31.875$.

$$x_{i+1} = f(x_i) = \begin{cases} x_i + [1]P_1 + [2^{S(x_i)/2}]P_2 & \text{if } S(x_i) \equiv 0 \pmod 2 \\ x_i + [1]P_1 + [-2^{(S(x_i)-1)/2}]P_2 & \text{if } S(x_i) \equiv 1 \pmod 2. \end{cases}$$

Pseudorandom Walk - 2D 8. The number of partitions of G are $n_s = 8$.

$m_1 = 1$ and $m_2 = 42.5$.

$$x_{i+1} = f(x_i) = \begin{cases} x_i + [1]P_1 + [2^{S(x_i)+1}]P_2 & \text{if } S(x_i) \equiv 0 \pmod 2 \\ x_i + [1]P_1 + [-2^{S(x_i)}]P_2 & \text{if } S(x_i) \equiv 1 \pmod 2. \end{cases}$$

Pseudorandom Walk - 2D 9. The number of partitions of G are $n_s = 16$. $m_1 = 1$ and $m_2 = 31.875$.

$$x_{i+1} = f(x_i) = x_i + [1]P_1 + [z_j]P_2$$

where the steps z_j for $0 \leq j \leq 15$ are uniformly distributed at random in the interval $[-2m_2, 2m_2]$. For our experiment we used Magma to produce the following the set of steps

$$\{-58, -54, -46, -45, -26, -14, -5, -2, 1, 3, 11, 24, 38, 56, 58, 62\}.$$

Pseudorandom Walk - 2D 10. The number of partitions of G are $n_s = 8$. $m_1 = 1$ and $m_2 = 42.5$.

$$x_{i+1} = f(x_i) = x_i + [1]P_1 + [y_j]P_2$$

where the steps z_j for $0 \leq j \leq 7$ are uniformly distributed at random in the interval $[-2m_2, 2m_2]$. For our experiment we used Magma to produce the following the set of steps

$$\{-73, -56, -27, -3, 7, 23, 69, 79\}.$$

Pseudorandom walks 6, 7 and 8 are of type '1-forward and side-to-side'. Pseudorandom walks 9 and 10 are of type '1-forward and side-to-side uniformly', with pseudorandom walk 9 being the walk that Gaudry and Schost [19] use. The differences between these walks are the mean absolute step size m_2, the number of partitions of G, n_s, and the way in which the steps sizes are constructed i.e. powers if a base b or predetermined random steps. To stop the walks visiting an element with the same exponents we always move in a positive direction in a particular component namely P_1 where the mean step size is 1 for all the walks. Pseudorandom walk 5 is a '1-forward and to the right' walk. There is no side-to-side aspect to this walk and in that sense this walk works very much like the walk in the 1-dimensional case.

Pseudorandom walk	5	6	7	8	9	10
n_s	8	8	16	8	16	8
Mean Step size in P_2 component	31.875	3.75	31.875	42.5	31.875	42.5
$x\times$ expected number of steps	1.16	1.69	1.10	1.20	1.09	1.16

Table 6.4: Average number of steps compared to the Birthday Paradox for the different 2D walks

Table 6.4 gives the results after averaging out over all the experiments for a particular walk. We can see that all the walks except pseudorandom walk 6, terminate on average close to the expected number of steps. A possible reason why pseudorandom walk 6, in particular, behaved worse on average than the other walks is that the mean absolute step size was small and so the walks were too localised. We can see that the walks of type '1-forward and side-to-side uniformly' (Pseudorandom walks 9 and 10) are marginally better than those of

type '1-forward and side-to-side' with the large mean step sizes (Pseudorandom walks 7 and 8). Also we found that although the '1-forward and to the right' walk was nearly as good as the side-to-side walks the area where walks could not start was far larger so this walk is less useful in practice.

6.4 d-dimensional discrete logarithm problem for $d \geq 3$

We have seen the improved Gaudry-Schost algorithm being used to solve both the 1-dimensional and 2-dimensional DLP. Now we consider solving the generic multidimensional DLP using this approach. However to compare the original and improved Gaudry-Schost algorithm when extended to d dimensions, it is easier to consider them in an idealised model. In this model we have only 1 processor and we pick elements from the tame and wild sets uniformly at random and store all the elements that are selected. Of course, the BSGS algorithm will be faster in this model but we are using this model to strictly compare the original and improved Gaudry-Schost algorithm.

Theorem 6.4.1. *Given the multidimensional discrete logarithm problem as described in Definition 6.0.1 let N be the cardinality of the search space as given by equation (6.2). Then the expected number of group operations (in the idealised model) in the worst case before we can solve for the d-dimensional discrete logarithm problem using the original Gaudry-Schost algorithm is*

$$2^{d/2}\sqrt{\pi N}, \tag{6.8}$$

and the average case expected number of group operations is

$$(4 - 2\sqrt{2})^d \sqrt{\pi N}. \tag{6.9}$$

Proof. In the worst case Q lies in a 'corner' of the search space and so the overlap between the original tame and wild sets has cardinality

$$M = \frac{N}{2^d}.$$

Therefore the expected number of steps before we expect to step in the overlap between the tame and wild sets is

$$\frac{N}{M} = 2^d.$$

So the expected number of group operations in the worst case is given by

$$2^d \sqrt{\pi M} = 2^{d/2} \sqrt{\pi N}$$

which proves the result presented in equation (6.8). To find the average case expected running time we have to average over all possible Q. At this point the proof is analogous to the 2-dimensional proof of Lemma 6.3.2. Without loss of generality let us consider problem instances in one 'corner' of the search space. Let

$$Q = [-N_1 + x_1 \tilde{N}_1]P_1 + [-N_2 + x_2 \tilde{N}_2]P_2 + \ldots + [-N_d + x_d \tilde{N}_d]P_d$$

for $x_i \in [0, 1/2]$. The size of the overlap, $\mathcal{A}$, between the original tame and wild

sets is

$$M = \left(\frac{1}{2} + x_1\right)\tilde{N}_1 \cdot \left(\frac{1}{2} + x_2\right)\tilde{N}_2 \cdots \left(\frac{1}{2} + x_d\right)\tilde{N}_d = N\prod_{i=1}^{d}\left(\frac{1}{2} + x_i\right)$$

So the expected number of steps a tame or wild walk need to take to step in $\mathcal{A}$ is

$$\frac{N}{M} = \prod_{i=1}^{d}\left(\frac{1}{2} + x_i\right)^{-1}$$

So the average case expected running time is like

$$\begin{aligned}
&2^d \int_{x_1=0}^{1/2}\int_{x_2=0}^{1/2}\cdots\int_{x_d=0}^{1/2}\prod_{i=1}^{d}\left(\frac{1}{2} + x_i\right)^{-1/2}\sqrt{\pi N}\, dx_1\, dx_2 \ldots dx_d \\
&= 2^d\sqrt{\pi N}\left(\int_{x=0}^{1/2}\left(\frac{1}{2} + x\right)^{-1/2} dx\right)^d \\
&= (4 - 2\sqrt{2})^d\sqrt{\pi N}
\end{aligned}$$

which proves the result presented in equation (6.9). □

In Sections 4.1.5 and 6.3.1.2 we improved upon the Gaudry-Schost algorithm in the 1-dimensional and 2-dimensional cases respectively by making the overlap, $\mathcal{A}$, between the tame and wild sets constant. Therefore we will continue by doing this for all dimensions.

Theorem 6.4.2. *Given the multidimensional discrete logarithm problem as described in Definition 6.0.1 let N be the cardinality of the search space as given by equation (6.2). Then the expected number of group operations (in the idealised model) before we can solve for the d-dimensional discrete logarithm problem using*

the improved Gaudry-Schost algorithm is

$$\frac{2^d}{3^{d/2}}\sqrt{\pi N}.$$

This is the expected running time in the best, worst and average cases.

Proof. Using the analogous strategy as that used in the previous dimensions we search in a hypercube of size $k^d N$ centred in the middle of the tame set and we search a space of the same size but split into 2^d disjoint sets in the 'corners' of the wild set. Each of these disjoint sets are of size

$$\left(\frac{k}{2}\right)^d N.$$

If we take $k = 2/3$ we can use the an analogous argument to that of Lemma 6.3.3 to show that the cardinality of the overlap, M', is constant for all problem instances. Simply as each dimension is independent of the others we can reduce back to the 1-dimensional case for each P_i component and therefore by taking $k = 2/3$ we can see that for all problem instances the cardinality of $\mathcal{A}$ is

$$M' = \frac{N}{3^d},$$

and the new search space is given by

$$N' = \frac{2^d N}{3^d}.$$

Therefore the expected number of steps before we expect to step in the overlap

between the tame and wild sets is

$$\frac{N'}{M'} = 2^d.$$

Using a Tame-Wild Birthday Paradox analysis, the number of group operations before we expect a collision is

$$2^d\sqrt{\pi M'} = 2^d\sqrt{\pi\frac{N}{3^d}} = \frac{2^d}{3^{d/2}}\sqrt{\pi N}.$$

□

We can use Theorems 6.4.1 and 6.4.2 to compare the original and improved Gaudry-Schost algorithm in both the worst and average cases. In the average case the improved Gaudry-Schost is expected to terminate in $1.15^d\sqrt{\pi N}$ group operations as opposed to $1.17^d\sqrt{\pi N}$ in the original algorithm. In the worst case the improvement is even more significant with $1.15^d\sqrt{\pi N}$ group operations in the improvement algorithm as opposed to $1.41^d\sqrt{\pi N}$ in the original algorithm. The improved Gaudry-Schost is clearly the algorithm of choice when solving the d-dimensional DLP for $d > 1$.

At this point all that remains to solve the multidimensional DLP is to pick a pseudorandom walk for the particular dimension d. However there is an issue that occurs with θ as d increases. The Gaudry-Schost algorithm expects to store approximately $K\theta$ distinguished points where K is the total number of steps. Using the Tame-Wild Birthday Paradox we know that in the multidimensional DLP $K = O(\sqrt{N})$, so to have only a constant storage requirement we have to take $\theta = \frac{c}{\sqrt{N}}$ where $c > 1$. Therefore we can expect to store close to c distinguished

points before we can solve for the multidimensional DLP. This is not a problem but what is an issue is that as we increase the dimension we have to consider the expected lengths of walks. The expected walk length is $\frac{m}{\theta}$ (Lemma 4.1.3), where m is the mean step size in any particular dimension, which means that a walk is of length $\frac{m\sqrt{N}}{c}$. This is, again, not a problem in the 1-dimensional case but in the 2-dimensional case if we say $\tilde{N}_1$ and $\tilde{N}_2$ are of the same order then each walk could traverse across the entire interval of the P_1 or P_2 components. In practice the constant c can be made fairly large (see Example 6.4.3) making this a non issue in the 2-dimensional case.

Example 6.4.3. As storage space is now very cheap, let our server be a standard PC with 1 terabyte or 2^{43} bits of hard disk space. If the cardinality of the group we are working in is $|G| = 2^{256}$ then in the 3-dimensional case we will be storing a 4-tuple. Each 4-tuple will use upto 256×4 bits of storage or 2^{10} bits. Therefore the number of distinguished points we can store is

$$\frac{2^{43}}{2^{10}} = 2^{33}.$$

Therefore in this case we could take $\theta = \frac{2^{33}}{\sqrt{N}}$.

However for $d \geq 3$ using $\theta = \frac{c}{\sqrt{N}}$ even for a large constant c can cause the majority of walks to step outside of the search space. For example if we say that $\tilde{N}_1, \tilde{N}_2$ and $\tilde{N}_3$ are of the same order then they are approximately $\sqrt[3]{N}$. Walks which have expected length $\frac{m\sqrt{N}}{c}$ will always walk outside of the search space so the Tame-Wild Birthday Paradox analysis on the size of the search space does not work. Relating this back to the pseudorandom walks, we cannot use walks which are always stepping in a positive direction in any dimensions. We need

to use walks which have a 'side-side' component in every dimension. Of course if c is large enough compared to the search space then this is not an issue but in general, as we want our algorithm to be low-memory we need to consider our pseudorandom walks carefully.

By having a 'side-to-side' component in every dimension, the boundaries (where walks are not permitted to start) from the edges will be defined using Lemma 5.1.1. In our setup the maximum excursion will be

$$\sqrt{\frac{2\sqrt{N}}{3c\pi}} = O(N^{1/4}). \tag{6.10}$$

As the interval of possible solutions in each dimension is $O(N^{1/3})$ these boundaries will fit well in our search space.

When $d = 4$ the value of the constant c does matter because if we take $\tilde{N_1}, \tilde{N_2}, \tilde{N_3}$ and $\tilde{N_4}$ to be approximately the same then they are of size $\sqrt[4]{N}$. Although the boundary from the edges where walks are not permitted to start is of order $O(N^{1/4})$ as c can be fairly large as shown in the Example 6.4.3 these boundaries will fit well in the search space.

For $d \geq 5$ the issue of boundaries reappears and this time cannot be resolved. The size of each dimension of the search space is smaller than the boundary given in equation (6.10) and as a result the entire idea of pseudorandom walks that pause at distinguished points fails if we have constant storage.

6.5 Conclusion

In this thesis we have considered variants of the DLP in particular the DLP in an interval and the multidimensional DLP. Whilst we have made some modest improvements to the Gaudry-Schost algorithm for solving the multidimensional DLP where $d > 1$ the most progress has been made in improving the Gaudry-Schost algorithm for solving the DLP in an interval.

We have presented the improved Pollard kangaroo algorithm which has yet to appear in the literature. Then by taking ideas from Pollard's new algorithm we have improved the Gaudry-Schost algorithm to make it the algorithm of choice for solving the DLP in an interval in a generic group. For groups with fast inversion we have made further improvements to the algorithm by using equivalence classes. They are many avenues with possibilities of future research that arise from this body of work. Below are some of the open problems.

- Can we model the 3 and 4 set Gaudry-Schost algorithm using a more accurate form of analysis than the Tame-Wild Birthday Paradox which takes into account a 3 set overlap, i.e. when $T \cap W_N \cap W_P$ is nonempty?
- For the Improved 4-set Gaudry-Schost algorithm is there a better strategy than searching only two thirds of the tame, wildn and wildp sets?
- Can we model the Gaudry-Schost algorithm using equivalence classes using a more accurate form of analysis which takes into account a different density of walks between the tame and wild sets?
- For the Improved Gaudry-Schost algorithm using equivalence classes is there

a better strategy than searching only one half of the wild set?

- When solving the multidimensional DLP where the possible solutions do not form a hypercube, is there a better approach than just to put a hypercube around the set of possible solutions and implementing the standard Gaudry-Schost algorithm?

- How do we overcome the boundary issues that arise in the multidimensional DLP for $d \geq 5$?

We hope that the algorithms presented in this work are of value to academics, researchers and those in industry who work in cryptography and information security.

Bibliography

[1] L. Adleman and J. Demarris. A Subexponential Alogrithm for Discrete Logarithms Over all Finite Fields. *Mathematics of Computation*, 61(203):1–15, 1994.

[2] W. Benits Jr. *Applications of Frobenius Expansions in Elliptic Curve Cryptography*. PhD thesis.

[3] R. Brent. An Improved Monte Carlo Factorization Algorithm. *BIT*, 20:176–184, 1980.

[4] D. R. L. Brown. Standards for Efficient Cryptography 1 (SEC 1): Elliptic Curve Cryptography. Technical report, Certicom Corporation, 2009.

[5] J. H. Cheon. Security Analysis of the Strong Diffie-Hellman Problem. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 2006.

[6] J. H. Cheon, J. Hong, and M. Kim. Speeding Up the Pollard Rho Method on Prime Fields. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 471–488. Springer-Verlag, 2008.

[7] E. G. Cofman, P. Flajolet, L. Flatto, and M. Hofri. The Maximum of a Random Walk and its Application to Rectangle Packing. Technical report, INRIA, 1997.

[8] W. Diffie and M. Hellman. New Directions in Cryptology. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[9] T. ElGamal. A Public key Cryptosystem and a Signature Scheme based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469 – 472, 1985.

[10] R. W. Floyd. Non-determintistic Algorithms. *Journal of the ACM*, 14(4):636–644, 1967.

[11] S. D. Galbraith. *Mathematics of Public Key Cryptography*. In preparation.

[12] S. D. Galbraith, X. Lin, and M. Scott. Endomorphisms for Faster Elliptic Curve Cryptography on a Large Class of Curves. In A. Joux, editor, *Eurocrypt 2009*, Lecture Notes in Computer Science. Springer-Verlag, 2009.

[13] S. D. Galbraith, J. M. Pollard, and R. S. Ruprai. Improving kangaroo and Gaudry-Schost methods for solving the DLP in an interval. In preparation.

[14] S. D. Galbraith and R. S. Ruprai. Faster Algorithms for the Discrete Logarithm Problem in a Short Interval. Resubmitted to the Public Key Cryptography 2010 conference in Paris, France.

[15] S. D. Galbraith and R. S. Ruprai. An Improvement to the Gaudry-Schost Algorithm for Multidimensional Discrete Logarithm Problems. In M. G.

Parker, editor, *Cryptography and Coding*, volume 5921 of *Lecture Notes in Computer Science*, pages 368–382. Springer-Verlag, 2009.

[16] S. D. Galbraith and M. Scott. Exponentiation in Pairing-Friendly Groups using Homomorphisms. In S. D. Galbraith and K. Patterson, editors, *Pairings 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 211–224. Springer-Verlag, 2008.

[17] R. Gallant, R. Lambert, and S. Vanstone. Improving the Parallelized Pollard Lambda Search on Binary Anomalous Curves. *Mathematics of Computation*, 69:1699–1705, 2000.

[18] R. P. Gallant, R. J. Lambert, and S. A. Vanstone. Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer-Verlag, 2001.

[19] P. Gaudry and E. Schost. A low-memory parallel version of Matsuo, Chao and Tsujii's algorithm. In *Proceedings of Algorithm Number Theory Symposium - ANTS VI*, volume 3076 of *Lecture Notes in Computer Science*, pages 208–222. Springer-Verlag, 2004.

[20] R. Gennaro. An Improved Pseudo-random Generator Based on Discrete Log. In *Crypto 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 469–481. Springer-Verlag, 2000.

[21] K. Gopalakrishnan, N. Thériault, and C. Z. Yao. Solving Discrete Logarithms from Partial Knowledge of the Key. In K. Srinathan, C. P. Rangan,

and M. Yung, editors, *Indocrypt*, volume 4859 of *Lecture Notes in Computer Science*, pages 224–237. Springer-Verlag, 2007.

[22] D. M. Gordon. Discrete Logarithms in $GF(p)$ using the Number Field Sieve. *SIAM Journal of Discrete Mathematics*, 6:124–138, 1992.

[23] U. S. Government. Digital signature standard. Technical report, Federal Information Processing Standards Publication 186, 1994.

[24] D. Jao and K. Yoshida. Boneh-Boyen signatures and the Strong Diffie-Hellman problem. In H. Shacham and B. Waters, editors, *Pairings 2009*, Lecture Notes in Computer Science. Springer-Verlag, 2009.

[25] A. Joux and R. Lercier. Improvements to the General Number Field Sieve for Discrete Logarithms in Prime Fields. A Comparison with the Gaussian Integer Method. *Mathematics of Computation*, 72(242):953–967, 2002.

[26] A. Joux, R. Lercier, N. Smart, and F. Vercauteren. The Number Field Sieve in the Medium Prime Case. In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 323–341. Springer-Verlag, 2006.

[27] D. E. Knuth. *Art of Computer Programming*, volume 2. Addision-Wesley, 3rd edition, 1997.

[28] N. Koblitz. CM-curves with Good Cryptographic Properties. In J. Feigenbaum, editor, *Advances in Cryptology - Crypto '91, 11th Annual International Cryptology Conference*, volume 576 of *Lecture Notes in Computer Science*, pages 279–287. Springer-Verlag, 1991.

[29] C. H. Lim and P. J. Lee. A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup. In B. S. Kaliski Jr., editor, *Crypto*, volume 1294 of *Lecture Notes in Computer Science*, pages 249–263. Springer-Verlag, 1997.

[30] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, 1996.

[31] K. Nishimura and M. Sibuya. Probability to meet in the middle. *Journal of Cryptology*, 2:13–22, 1990.

[32] G. Nivasch. Cycle detection using a stack. *Information Processing Letters*, 90(3):135–140, 2004.

[33] S. Patel and G. Sundaram. An Efficient Discrete Log Pseudo Random Generator. In *Crypto '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 304–317. Springer-Verlag, 1998.

[34] J. M. Pollard. Monte Carlo Methods for Index Computation mod p. *Mathematics of Computation*, 32(143):918–924, 1978.

[35] J. M. Pollard. Kangaroos, Monopoly and Discrete Logarithms. *Journal of Cryptology*, 13:437–447, 2000.

[36] J. M. Pollard. Three kangaroos are better than two! Private Communication, April 2009.

[37] R. S. Ruprai. Elliptic Curve Cryptography. Master's thesis, London School of Economics, University of London, 2005/6.

[38] O. Schirokauer, D. Weber, and T. Denny. Discrete Logarithms: The Effectiveness of the Index Calculus Method.

[39] R. Sedgewick, T. Szymanski, and A. Chi-Chih Yao. The Complexity of Finding Cycles in Periodic Functions. *SIAM Journal of Computation*, 11(2):376–390, 1982.

[40] D. Shanks. Class number, a Theory of Factorization and Genera. In *Proc. Symposium in Pure Mathematics*, volume 20, pages 415–440, 1971.

[41] N. Smart. *Cryptography: An Introduction*. Mcgraw-Hill, 2004.

[42] D. Stinson. Some Baby-Step Giant-Step Algorithms for Low Hamming Weight Discrete Logarithm Problem. *Mathematics of Computation*, 71(237):379–391, 2002.

[43] D. Stinson. *Cryptography: Theory and Practice*. Chapman & Hall/CRC, 3rd edition, 2006.

[44] E. Teske. Speeding up Pollard's rho Method for Computing Discrete Logarithms. In J. Buhler, editor, *ANTS 1998*, volume 1423 of *Lecture Notes in Computer Science*, pages 541–554. Springer-Verlag, 1998.

[45] E. Teske. Computing Discrete Logarithms with the Parallelized Kangaroo Method. *Discrete Applied Mathematics*, 130:61–82, 2003.

[46] P. C. van Oorschot and M. J. Wiener. Parallel collision Search with Application to Hash Functions and Discrete Logarithms. In *2nd ACM Conference on Computer and Communications Security*, pages 210–218, 1994.

[47] P. C. van Oorschot and M. J. Wiener. On Diffie-Hellman Key Agreement with Short Exponents. In *Eurocrypt*, volume 1070 of *Lecture Notes in Computer Science*, pages 332–343. Springer-Verlag, 1996.

[48] P. C. van Oorschot and M. J. Wiener. Parallel collision Search with Cryptanalytic Applications. *Journal of Cryptology*, 12:1–28, 1999.

[49] A. Weil. Numbers of Solutions of Equations in Finite Fields. *Bulletin of the American Mathematical Society*, 55:497–508, 1949.

[50] M. J. Wiener and R. J. Zuccerato. Faster Attacks on Elliptic Curve Cryptosystems. In S. E. Tavares and H. Meijer, editors, *Selected Areas in Cryptography*, volume 1556 of *Lecture Notes in Computer Science*, pages 190–200. Springer-Verlag, 1998.