Algorithms for the Elliptic Curve Discrete Logarithm Problem and the Approximate Common Divisor Problem

Shishay Welay Gebregiyorgis

A Thesis Submitted in Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Mathematics

The University of Auckland

February 2016

Abstract

Public key cryptosystems such as Diffie-Hellman key exchange and homomorphic encryption over the integers are based on the assumption that the Discrete Logarithm Problem (DLP) and the Approximate Common Divisor (ACD) problem are hard respectively. These computational assumptions can be tested by developing improved algorithms to solve them.

The DLP for elliptic curves defined over certain finite fields is believed to be hard. The best current algorithm for this problem is Pollard rho. The most promising new idea for attacking the DLP over these curves is the index calculus algorithm, since it solves the DLP for finite fields in subexponential time. It is important to understand this class of algorithms. We study the index calculus algorithm based on summation polynomials. This reduces the DLP to solving systems of multivariate polynomial equations. We explain recent research on exploiting symmetries arising from points of small order. The use of such symmetries can be used to speed up solving the system of polynomial equations, and hence speed up the algorithm.

We give an improved index calculus algorithm for solving the DLP for binary elliptic curves. Despite our improved ideas, our experiments suggest that Pollard rho is still the best algorithm for the DLP in practice. We discuss and analyse a new idea called the "splitting technique", which does not make use of symmetries. We finally suggest a new definition of the factor base to bring the probability of finding a relation close to 1.

To extend the notion of symmetries we investigate the use of an automorphism of elliptic curves defined over a field of characteristic 3 to speed up the index calculus algorithm. Our finding is that an automorphism speeds up the algorithm, but not to the extent that we would wish.

Finally we review, compare and precisely analyse some existing algorithms to solve the ACD problem. Our experiments show that the Cohn-Heninger algorithm is slower than the orthogonal lattice based approach. We propose a preprocessing of the ACD instances to speed up these algorithms. We explain that the preprocessing does not seem to threaten the ACD problem in practice.

Acknowledgements

I like to thank to my PhD supervisor Steven Galbraith for supporting me during my three years stay in the University of Auckland. He did not only supervise me but also he taught me how to be an independent self confident researcher. Above all, special appreciation goes to him for our collaborative work in my first published paper on the results in Chapter 3.

I also like to thank Arkadii Slinko, Ben Martin, Igor Klep for their comments and suggestions.

My work has been supported by the University of Auckland. The University of Auckland did not only provide me materials needed for my PhD program but also for awarding me a University of Auckland Doctoral Scholarship. So I really like to thank to all staff in this great University.

Finally huge thanks goes to my family for supporting me in all ways. All the work is dedicated to my amazing family.

Contents

THE UNIVERSITY OF AUCKLAND
SCHOOL OF GRADUATE STUDIES

Co-Authorship Form

Graduate Centre
ClockTower – East Wing
22 Princes Street, Auckland
Phone: +64 9 373 7599 ext 81321
Fax: +64 9 373 7610
Email: postgraduate@auckland.ac.nz
www.postgrad.auckland.ac.nz

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

Chapter 3

Nature of contribution by PhD candidate	Analysis of ideas, computer experiments, writing
Extent of contribution by PhD candidate (%)	80

CO-AUTHORS

Name	Nature of Contribution
S. Galbraith	Initial ideas, suggestions and advice

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- in cases where the PhD candidate was the lead author of the work that the candidate wrote the text.

Name	Signature	Date
Steven Galbraith	S Galbraith	14-9-2015

Chapter 1

Cryptography and Computational Assumptions

Contents

Secure cryptosystems are built using computational problems that are believed to be hard. The RSA and Diffie-Hellman key exchange are based on the assumption that the integer factorization and discrete logarithm problems are hard respectively. If we can break the underlying assumption, then the cryptosystem is not secure any more. In this regard, we are interested in trying to solve the underlying hard computational problems of cryptosystems. If the computational problem is intrinsically easy, we can provide an algorithm to solve the problem in polynomial time. If however the computational problem is intrinsically difficult, we would wish to show that there is no algorithm that solves the problem. The lack of proof showing that there is no efficient algorithm to solve the underlying hard computational problems in many cryptosytems is our motivation for our research. We test computational assumptions by developing improved algorithms to solve them. We give a summary of the existing algorithms to solve the integer factorization and discrete logarithm problems.

1.1 Cryptography

Cryptography deals with securing communication channels, electronic transactions, sensitive data and other critical information such as medical records. It is concerned with designing a cyptosystem or a cryptographic system that is capable of providing services likes confidentiality, authenticity, integrity and non-repudiation. By confidentiality we mean that a cryptosystem is intended to allow access to sensitive data or information only to authorized users. Authenticity refers to the ability of a cryptosystem to validate the source of data origin. Integrity refers to the assurance of a cryptosystem that a message was not modified in transit intentionally or unintentionally through insertions, deletion, or modification. Non-repudation refers to the ability of the cryptosystem to provide evidence in case a dispute arises by a sender claiming that he/she did not send the data.

Nowadays, our daily secure digital communications are involving a cryptosystem. For example in cellular communications, internet and browsers. More interestingly, the heart of the digital currency Bitcoin [Nak09] is a cryptosystem. So cryptography plays a great role in our daily lives.

There are two branches of cryptography, namely symmetric and asymmetric. Symmetric cryptography deals with designing cryptosystems (encryption functions, cryptographic hash functions, message authentication codes etc.) based on traditional design methodologies under the assumption that a shared key, which is an essential part of a cryptosystem, is available between communicating parties. A symmetric cryptosystem is composed of two transformation functions, an encryption $\mathcal{E}_k$ and a decryption function $\mathcal{D}_k$. The encryption function $\mathcal{E}_k$ takes a message m and a key k and produces a ciphertext c. The decryption function does the inverse, that is it takes a ciphertext c and a key k and recovers the original message m. The two functions are assumed to be available publicly. If two communicating parties Alice and Bob want to communicate securely under this cryptosystem, they need to agree on a common key K beforehand in a safe way. If the key k is compromised then the cryptosystem gives no security. The main drawback of A symmetric cryptography is key management and distribution.

Asymmetric cryptography, known as public key cryptography, uses hard computational problems to design a cryptosystem, which allows two or more parties to communicate securely over unsecured channels without the need for prior key agreement. There are several functionalities provided by public key cryptography such as encryption, signatures, key exchange and identification services.

In public key cryptography, two keys are generated. A private key s_k and a public key p_k. Every user will have two keys. The private key is kept secret whereas the public key is published in a directory so that any one else has access to it. The computational assumption is that given the public key p_k, it is hard to determine the private key s_k.

If Alice wants to send a sensitive message to Bob over an insecure channel, Alice obtains the public key p_k of Bob and uses it to encrypt the message. Bob upon receiving the encrypted message uses his private key s_k to decrypt the message. We require the encryption function to be easily computable but hard to invert. Functions which are easy to compute but hard to invert are called one-way functions.

Definition 1.1.1. *(One-way functions) Let k be a security parameter and n be a function of k. Let f be $f : \{0,1\}^* \mapsto \{0,1\}^*$. Then f is a one-way function if*

1. *f is easy to compute. For all n and $x \in \{0,1\}^n$, there is a deterministic polynomial time algorithm f_{eval} such that $f_{\text{eval}}(x) = f(x)$.*
2. *f is hard to invert. For all probabilistic polynomial time algorithms A,*

$$\Pr\Big[x \leftarrow \{0,1\}^n, y = f(x), x' \leftarrow A(1^n, y) \mid f(x') = y\Big] < \frac{1}{2^k}.$$

In addition to the one-wayness property of the encryption function, we also require Bob, who posses the private key s_k to decrypt the message. So we allow the decryption to be possible using a trapdoor, a secret information that allows to easily invert the encryption function. Such functions are called one-way trapdoor functions.

These one-way trapdoor functions are the building blocks of modern cryptosystems based on computational number theoretic assumptions such as the integer factorization and discrete logarithm problems.

1.2 The integer factorization problem and RSA cryptosystem

Definition 1.2.1. *(Integer Factorization Problem) Let N be a positive composite integer. The integer factorization problem is to find the prime factorization of N and write N as*

$$N = \prod_{i=1}^{k} p_i^{e_i},$$

where $e_i \geq 1$ and the primes p_i are pair-wise co-prime.

The integer factorization problem is a well-studied problem and it is one of the number theoretic computational problems believed to be a candidate for one-way function. The RSA [RSA78] public key cryptosystem is based on the intractability assumption of factoring a large composite integer $N = pq$, where p and q are two distinct primes. The integer N is called modulus.

The "naive" RSA cryptosystem is composed of the following algorithmic functions (See Chapter 24 Section 1 page 486 of [Gal12]).

KeyGen(k): On input a security parameter k, the probabilistic polynomial time key generation algorithm KeyGen(k) generates two distinct primes p and q of size approximately $k/2$ bits each and sets $N = pq$. It chooses a random integer e co-prime to $p - 1$ and $q - 1$ such that $p, q \not\equiv 1 \pmod{e}$, and computes $d = e^{-1} \pmod{\lambda(N)}$, where $\lambda(N) = \text{LCM}(p - 1, q - 1)$ is the Carmichael lambda function. The key generation algorithm then outputs $s_k = (N, d)$ as the private key and $p_k = (N, e)$ as the public key. Note that LCM stands for least common multiple.

Encrypt(p_k, m): Let $\mathcal{P} = \mathcal{C} = \mathbb{Z}_N^*$ be the plaintext and ciphertext spaces. The polynomial time encryption algorithm Encrypt(p_k, m) takes the public key p_k, and $m \in \mathcal{P}$ as input and outputs $c = m^e \pmod{N}$, where $c \in \mathcal{C}$ is the encryption of the message m.

Decrypt(c, s_k): The deterministic polynomial time decryption algorithm Decrypt(c, s_k) takes the private key s_k and the cipher text c as input and outputs $m = c^d \pmod{N}$. It is required that Decrypt(Encrypt$(p_k, m), s_k) = m$.

Sign(m, s_k): On input the private key parameter s_k and a message m. The probabilistic polynomial time signing algorithm Sign(m, s_k) outputs $s = m^d \pmod{N}$, a signature of the message m. Note that there attacks such as a chosen-plaintext and a chosen-ciphertext against this plain signature scheme. An actual signature algorithm uses padding schemes and hash functions.

Verify(m, s, p_k): On input the public key parameter p_k, the signature s, and the message m, the deterministic polynomial time verification algorithm Verify(m, s, p_k) computes $\tilde{m} \equiv s^e \pmod{n}$ and outputs valid if $\tilde{m} = m$ otherwise returns invalid.

For encryption algorithm to be fast, it is tempting to take the public key e to be small such as $e \in \{2^4 + 1, 2^{16} + 1\}$. The Rabin cryptosystem is a special type of RSA cryptosystem with $e = 2$ and the primes p and q are selected to satisfy $p \equiv q \equiv 3 \pmod 4$ to simplify computations. We refer to Chapter 24 Section 2 page 491 of [Gal12] for details.

To encrypt a message m given the public parameters $p_k = (N, e)$, we compute $c \equiv m^e \pmod N$. The corresponding private key $s_k = (N, d)$ acts as a trapdoor. Where as in the signature scheme, the holder of the private key parameters signs a message or a document using the private key parameters. One can then verify that indeed the message or document is signed by who claim to be the legitimate signer.

Definition 1.2.2. *(RSA Problem) Let $c \equiv m^e \pmod N$, where $N = pq$ is a product of two distinct primes. The RSA problem is to recover m given c and the public key parameters $p_k = (N, e)$. In other words, the RSA problem is computing the e^{th} root modulo N.*

If factoring is easy, clearly breaking the RSA cryptosystem is easy too. We factor N to get its two prime factors p and q, and we compute $\lambda(N) = \text{LCM}(p-1, q-1)$. Finally we recover d by computing $e^{-1} \pmod{\lambda(N)}$ using extended euclidean algorithm. So in order for the RSA cryptosystem to be secure, p and q should be large primes such that it is computational infeasible to factor N with current methods.

1.3 Integer factorization algorithms

1.3.1 Pollard $p-1$ algorithm

Definition 1.3.1. (B-Smooth) Let $N = \prod_{i=1}^{r} p_i^{e_i}$ be a positive integer, where the p_i are distinct primes and $e_i \geq 1$. Let B be some positive integer. If $p_i \leq B$ for $1 \leq i \leq r$, then N is called B-Smooth and if $p_i^{e_i} \leq B$ for $1 \leq i \leq r$, then N is called B-Power Smooth.

Let p be a prime divisor of N and B be a smoothness bound. The idea behind the Pollard $p-1$ algorithm [Pol74] is if $p-1$ is B-Power Smooth, then we can find a non-trivial factor of N. Indeed if $p-1$ is B-Power Smooth, then $(p-1) \mid B!$. We refer to Chapter 12 Section 3 of the book [Gal12] and Chapter 5 Section 6 of the book [Sti56] for reference.

Let $a \in \mathbb{Z}/N\mathbb{Z}$ be a random element. Suppose $b = a$, the Pollard $p-1$ algorithm iteratively computes

$$b \leftarrow b^j \pmod N \quad \text{for} \quad 2 \leq j \leq B.$$

At the end of the iteration, we observe that $b \equiv a^{B!} \pmod N$. Since $p \mid N$, we have $b \equiv a^{B!} \pmod p$. By Fermat's little theorem $a^{B!} \equiv 1 \pmod p$ and hence,

$$b \equiv a^{B!} \equiv 1 \pmod p \implies p \mid (b-1).$$

Since p divides both N and $b-1$, with high probability we can find a non-trivial factor $d \neq \{1, N\}$ by computing

$$d = \text{GCD}(b-1, N).$$

Lemma 1.3.2. *Let N be an odd composite integer and B be a bound for smoothness. Then the Pollard $p-1$ factorization algorithm has a total complexity of*

$$O\Big(B \log B (\log N)^2 + (\log N)^3\Big)$$

bit operations.

The running time of the Pollard $p-1$ algorithm is exponential in B. So it is effective for small bound B. This restricts N to have a prime factor p such that $p-1$ has small prime factors. This makes it impractical for factoring an RSA modulus N.

1.3.2 The elliptic curve factorization method

The elliptic curve factorization method [Len87] uses the same concepts as the Pollard $p-1$ factorization algorithm. Instead of working with the group $\mathbb{Z}_N^*$ as in the Pollard $p-1$ factorization algorithm, we work over an elliptic curve. The requirement that $p-1$ is B-Smooth is also relaxed with this method.

Let $N = pq$ where p and q are prime factors, the elliptic curve factorization method proceeds by randomly choosing x_1, y_1, and a from the set $\{2, \cdots, N-1\}$ to form a random elliptic curve E (see Chapter 2 Section 2.2 for elliptic curve definition)

$$E : y^2 z = x^3 + axz^2 + bz^3$$

over $\mathbb{Z}/N\mathbb{Z}$ such that $b = y_1^2 - x_1^3 - ax_1 \pmod N$. Note that by the Chinese remainder theorem $E(\mathbb{Z}_N) \equiv E(\mathbb{F}_p) \times E(\mathbb{F}_q)$.

We observe that $P = (x_1 : y_1 : 1)$ is a point on the elliptic curve E. The algorithm sets $Q = P$ and iteratively computes

$$Q \leftarrow [j]Q \quad \text{for} \quad 2 \leq j \leq B.$$

At the end of the iteration, we get $Q = [B!]P \in E(\mathbb{F}_p)$. We hope $\#E(\mathbb{F}_p)$ to be B-Smooth so that $\#E(\mathbb{F}_p) \mid B!$ which implies Q will be the identity element $(0 : 1 : 0)$. Since the z coordinate is zero, it must be the case that $p \mid z$. With high probability computing GCD(z, N) gives a non-trivial factor of N, where GCD stands for greatest common divisor.

Lemma 1.3.3. *Let N be an odd composite positive integer and denote by p the smallest prime factor of N. The elliptic curve factorization method has an asymptotic complexity of*

$$O\left(e^{(1+o(1))\sqrt{2\ln p \ln\ln p}}(\log N)^2\right)$$

bit operations.

Unlike the Pollard $p-1$ algorithm, if the elliptic factorization method fails, we can pick a different elliptic curve and it is likely that eventually $\#E(\mathbb{F}_p)$ is B-smooth.

1.3.3 The quadratic sieve factorization method

Let N be an RSA modulus that we like to factor. The idea of the quadratic sieve factorization algorithm [CP05] is based on the observation, if $x^2 \equiv y^2 \pmod N$ such that $x \neq \pm y$, then GCD$(x - y, N)$ and GCD$(x + y, N)$ are non-trivial factors of N. In this case, we have

$$(x-y)(x+y) \equiv 0 \pmod N \implies N \mid (x-y)(x+y).$$

Note that N neither divides $x - y$ nor $x + y$ as $x \neq \pm y$.

The quadratic sieve factorization method fixes a smoothness bound B and a sieving interval $[\lfloor\sqrt{N}\rfloor - M, \lfloor\sqrt{N}\rfloor + M]$ for some fixed positive integer M. Primes less than or equal to the bound B form a factor base $\mathcal{F}$.

Define the polynomial $Q(\tilde{x})$ to be $Q(\tilde{x}) = \tilde{x}^2 - N$ (see [Lan01]), then for x_1 in the sieving interval, $Q(x_1) \equiv x_1^2 \pmod N$. So if we compute $Q(x_1), Q(x_2), \cdots, Q(x_k)$ such that $Q(x_i)$ is B-Smooth for each x_i in the sieving interval, we have

$$Q(x_1)Q(x_2)\cdots Q(x_k) \equiv x_1^2 x_2^2 \cdots x_k^2 \pmod N. \tag{1.1}$$

To make the left side of equation (1.1) a square, the quadratic sieve factorization method finds a subset $Q(x_1), Q(x_2), \cdots, Q(x_r)$ such that the product $Q(x_1)Q(x_2)\cdots Q(x_r)$ is a square. Let $Q(x_1)Q(x_2)\cdots Q(x_r) = y^2$ and $x = x_1 x_2 \cdots x_r$. Then

$$Q(x_1)Q(x_2)\cdots Q(x_r) \equiv x_1^2 x_2^2 \cdots x_r^2 \pmod N \implies y^2 \equiv x^2 \pmod N.$$

Determining the subset $\{Q(x_1), Q(x_2), \cdots, Q(x_r)\}$ such that the product is a square can be achieved using a linear algebra. We collect relations of the form $Q(\tilde{x}) = \tilde{x}^2 - N$ for $\tilde{x}$ in the sieving interval which split over the factor base completely as $Q(\tilde{x}) = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, where p_i are primes constituting the factor base and $e_i \geq 0$ is the exponent. The relations are written as row vectors with entries $e_i \pmod 2$ to finally form a matrix A. Note if a product of a subset of the set $\{Q(x_1), Q(x_2), \cdots, Q(x_k)\}$ results in an exponent vector with all even entries, then the product is a perfect square. Thus finding the kernel of the matrix A modulo 2 gives the required set $\{Q(x_1), Q(x_2), \cdots, Q(x_r)\}$.

To reduce the size of the factor base and hence improve the linear algebra complexity, we can put a further restriction on the factor base elements. If $p_i \mid Q(\tilde{x})$ then $\tilde{x}^2 \equiv N \pmod{p_i}$ and so N is a quadratic residue modulo p_i. Choosing the factor base elements such that N is a quadratic residue modulo p_i gives a reduced size.

Lemma 1.3.4. *The quadratic sieve factorization algorithm has an asymptotic complexity of*

$$O\left(e^{(1+o(1))\sqrt{\ln N \ln \ln N}} (\log N)^2\right)$$

bit operations.

1.4 The discrete logarithm problem and Elgamal cryptosystem

As with the integer factorization problem, the discrete logarithm problem over finite fields is believed to be a hard computational problem.

Definition 1.4.1. *(DLP) Let $(G, .)$ be a multiplicative group. Let $g, h \in G$. Define $\langle g \rangle$ to be*

$$\langle g \rangle = \{g^i \mid 0 \leq i \leq r-1\},$$

where r is the order of g. Given $h \in \langle g \rangle$, the DLP is to find the unique integer a, $0 \leq a \leq r-1$, such that $h \equiv g^a$. It is denoted as $\log_g h$.

There are many cryptosystems that make use of the hardness of the discrete logarithm problem. A popular example is the Elgamal cryptosystem [Elg85] (see also Chapter 6 Section 1 of [Sti56] and Chapter 20 Section 3 of [Gal12]). As in the design motivation of the RSA and Rabin cryptosystems, the principle behind the design of DLP based cryptosystems is based on the one-way property of the exponentiation function. Given a, we can compute g^a easily using the square-and-multiply algorithm, whereas computing the inverse ($\log_g h$) is a difficult computational problem for appropriate size parameters. Currently a key length of 224 bits and a group size of 2048 bits is recommended [Gir15].

Let $G = \mathbb{Z}_p^*$, where p is a prime. The "naive" Elgamal cryptosystem is composed of the following algorithms.

KeyGen(k): Let k be a security parameter. On input k, the probabilistic polynomial time key generation algorithm KeyGen(k) generates a k bit prime p and an element $g \in \mathbb{Z}_p^*$. It then chooses a random integer $1 < a < p-1$ and sets $h \equiv g^a \pmod p$. The key generation algorithm finally outputs $s_k = (p, g, a)$ as the private key and $p_k = (p, g, h)$ as the public key.

Encrypt(p_k, m): Let $\mathcal{P} = \mathbb{Z}_p^*$ and $\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ be the plaintext and ciphertext spaces respectively. The probabilistic polynomial time encryption algorithm Encrypt(p_k, m) takes the public key p_k and a message $m \in \mathcal{P}$ as input. It chooses a random integer $1 < b < p-1$ and sets $c_1 \equiv g^b \pmod p$. It then outputs the ciphertext pairs $c = (c_1, c_2) \in \mathcal{C}$ as the encryption of the message m, where $c_2 \equiv mh^b \pmod p$.

Decrypt(c, s_k): The deterministic polynomial time decryption algorithm Decrypt(c, s_k) takes the private key s_k and the cipher text $c = (c_1, c_2)$ as input and outputs $m \equiv c_2 c_1^{-a} \pmod{p}$.

Given a ciphertext $c = (c_1, c_2)$, the Elgamal decryption correctly decrypts. Indeed

$$\begin{aligned} m &\equiv c_2 c_1^{-a} \pmod{p} \\ &\equiv m h^b g^{-ab} \pmod{p} \\ &\equiv m g^{ab} g^{-ab} \pmod{p} \\ &\equiv m \pmod{p}. \end{aligned}$$

If we are able to solve the discrete logarithm problem, then the Elgamal cryptosystem is not secure. For example, given a ciphertext $c = (c_1, c_2)$, if we can compute $\log_g c_1$ to recover b, then $c_2 h^{-b}$ gives the message m. We can also recover the private key a by computing $\log_g h$ from the public key parameter h to finally recover the message m by computing $m \equiv c_2 c_1^{-a} \pmod{p}$.

The Diffie-Hellman key exchange protocol [DH76] is another popular cryptosystem that makes use of the hardness of the DLP over G. Similar to the Elgamal cryptosystem, we will have common public key parameters p and g. Assume Alice and Bob want to exchange a key K over an insecure channel. Alice generates a random integer $1 < a < p - 1$ and sends $h_a \equiv g^a \pmod{p}$ to Bob. And Bob does the same, he chooses a random integer $1 < b < p - 1$ and sends $h_b \equiv g^b \pmod{p}$ to Alice. Then Alice and Bob compute the same key $K \equiv h_b^a \pmod{p}$ and $K \equiv h_a^b \pmod{p}$ respectively. The exchanged key K can be compromised if the Computational Diffie-Hellman (CDH) problem can be solved.

Definition 1.4.2. *(CDH) Let $G = \mathbb{Z}_p^*$ be a multiplicative group. Then the CDH problem is given the triple (g, g^a, g^b) elements of G to compute g^{ab}.*

As in the Elgamal cryptosystem, if we are able to solve the DLP, then clearly the CDH problem can be solved.

1.5 Algorithms for solving the discrete logarithm problem

Let $G = \mathbb{F}_p^*$ such that $G = \langle g \rangle$, where p is a prime. Let $h \in \langle g \rangle$. The DLP is to find a unique integer $a < p$ such that $h \equiv g^a$. The problem definition can also be extended to the case when $G \subseteq \mathbb{F}_q^*$ having a generator g with order r, where q is a prime power and $r = \#G$. We list some algorithms to solve the DLP problem.

1.5.1 The baby-step-giant-step algorithm

Let $m = \lceil \sqrt{r} \rceil$. Write a in base-m representation as $a = im + j$ for $0 \leq i, j \leq m$. The baby-step-giant-step algorithm [Sha71] is based on the observation $g^j = g^a (g^{-m})^i = h(g^{-m})^i$. The algorithm involves two steps.

The baby-step phase: For $0 \leq j < m$, g^j is computed and the values (j, g^j) are stored in a list.

The giant-step phase: For $0 \leq i < m$, $h(g^{-m})^i$ is computed and checked for a match in the second entry of the baby-step stored list.

If a match is found then $a = j + im$ is a solution to the DLP. The baby-step-giant-step is a typical example of a time/memory tradeoff algorithm. It requires $O(\sqrt{r})$ storage and $O(\sqrt{r})$ arithmetic operations in G.

1.5.2 The Pohlig-Hellman algorithm

Let the order of g be given as $r = \prod_{i=1}^{k} p_i^{e_i}$. The idea of Pohlig-Hellman algorithm [PH78] is based on the observation of the group homomorphism

$$\phi_{p_i}(g) = g^{r/(p_i^{e_i})}$$

from $\langle g \rangle$ to a cyclic subgroup of order p^e. Under the homomorphism, we have

$$\phi_{p_i}(h) \equiv \phi_{p_i}(g)^a \pmod{p_i^{e_i}}.$$

Let g_0 have order p^e for some $e > 1$ and let $h_0 \equiv g_0^a \pmod{r}$. Observe that a can be written as $a = a_0 + a_1 p + \cdots + a_{e-1}p^{e-1}$, where $0 \le a_i < p$. To compute a modulo p^e, it is enough to recover $(a_0, a_1, \cdots, a_{e-1})$.

Let $g_1 = g_0^{p^{e-1}}$. Then $h_0^{p^{e-1}} = g_1^{a_0}$. So we can recover a_0 by trying all possibilities (assuming p is not large). We can also use other methods such as the baby-step-giant-step algorithm.

To recover a_1, first define h_1 to be $h_1 = h_0 g_0^{-a_0} = g_0^{a_1 p + \cdots + a_{e-1}p^{e-1}}$. Then $h_1^{p^{e-2}} = g_1^{a_1}$. So we can recover a_1 using the same technique we have recovered a_0.

Similarly to recover a_2, we define $h_2 = h_1 g_0^{-a_1 p} = g_0^{a_2 p^2 + \cdots + a_{e-1}p^{e-1}}$. Then we observe that $h_2^{p^{e-3}} = g_1^{a_2}$ and a_2 can be recovered. We continue to recover all a_i this way. So now we know a modulo p^e. For each prime power $p_i^{e_i}$ in the factorization of r, compute a modulo $p_i^{e_i}$. Then using the Chinese remainder theorem, a can be recovered.

Lemma 1.5.1. *Let $g \in G$ has order r. Let B be a bound where r is $B-$*smooth. Then the DLP can be solved using the Pohlig-Hellman algorithm in

$$O\Big((\log r)^2 + B \log r\Big)$$

group arithmetic operations in G.

1.5.3 The Pollard rho algorithm

To compute $\log_g h$, the idea of Pollard rho algorithm [Pol78] is based on finding collisions between two sequences. Specifically, if we can find $(b_i, c_i, b_j, c_j) \in \mathbb{Z}/r\mathbb{Z}$ such that

$$g^{b_i} h^{c_i} = g^{b_j} h^{c_j},$$

where $c_i \not\equiv c_j \pmod{r}$, then $h = g^{(b_j - b_i)(c_i - c_j)^{-1} \pmod{r}}$. So $a \equiv (b_j - b_i)(c_i - c_j)^{-1} \pmod{r}$ is a solution to the DLP.

We start by partitioning the group G into three parts S_1, S_2, S_3 such that they are almost equal size. Define the pseudorandom function $f : (\langle g \rangle, \mathbb{Z}/r\mathbb{Z}, \mathbb{Z}/r\mathbb{Z}) \mapsto (\langle g \rangle, \mathbb{Z}/r\mathbb{Z}, \mathbb{Z}/r\mathbb{Z})$ as follows

$$(X_{i+1}, b_{i+1}, c_{i+1}) = f(X_i, b_i, c_i) = \begin{cases} (gX_i, b_i + 1, c_i) & \text{if } X_i \in S_1 \\ (X_i^2, 2b_i, 2c_i) & \text{if } X_i \in S_2 \\ (hX_i, b_i, c_i + 1) & \text{if } X_i \in S_3. \end{cases}$$

Let $X_i = g^{b_i} h^{c_i}$. The starting sequence (X_i, b_i, c_i) is defined to be $(1, 0, 0)$. Then we generate the next sequence $(X_{i+1}, b_{i+1}, c_{i+1})$ according to f. As G is a finite set, by birthday paradox we expect a collision $X_i = X_j$ for some $1 \le i < j$. If this is the case $g^{b_i} h^{c_i} = g^{b_j} h^{c_j}$. It is then immediate to find a solution to the DLP.

The DLP in group G can be solved heuristically using the Pollard rho algorithm in $O\Big(\sqrt{r}(\log r)^2\Big)$ group arithmetic operations in G.

1.5.4 The index calculus method

The index calculus method [AD94] is the most effective algorithm to solve the DLP in finite fields in subexponential running time. The concept is similar to the quadratic sieve factorization method (see Section 1.3.3). The index calculus method remains our motivation to provide an index calculus algorithm for solving discrete logarithm problem over binary curves presented in Chapter 3.

Let B be the smoothness bound. We define a factor base $\mathcal{F}$ to be the set of primes p_i less than B,

$$\mathcal{F} = \{p_1, p_2, \cdots, p_k\} \subset G.$$

The idea of the index calculus method is to find the discrete logarithm of each prime element in the factor base $\mathcal{F}$ with respect to g to finally solve the discrete logarithm problem. It has three stages.

Stage 1: This is called the relation generation stage. Pick a random value $k \in \mathbb{Z}/r\mathbb{Z}$ and compute $R \equiv g^k$. If R completely factors over the factor base $\mathcal{F}$, that is $R = \prod_{i=1}^{m} p_i^{e_i}$, then we have a relation. Taking logs, each relation gives $k \equiv \sum_{i=1}^{\#\mathcal{F}} e_i \log_g p_i \pmod{r}$. We store k as column vector and $(e_1, e_2, \cdots, e_{\#\mathcal{F}})$ as a row in a matrix.

Stage 2: This is a linear algebra stage. In this stage we compute the discrete logarithm of each factor base element with respect to g. If we collect $\#\mathcal{F}$ independent relations in stage 1, then applying Gaussian elimination on the full rank matrix gives actual values of the discrete logarithm of the factor base elements with respect to g. Note that if r is not prime, the Gaussian elimination (over the ring $\mathbb{Z}_r$) is not guaranteed to work.

Stage 3: In this stage we find a relation that includes h. We continuously pick a random value $k' \in \mathbb{Z}/r\mathbb{Z}$ until $hg^{k'}$ factors over the factor base $\mathcal{F}$. Assume $hg^{k'}$ factors over the factor base, $hg^{k'} \equiv \prod_{i=1}^{\#\mathcal{F}} p_i^{e'_i}$. Let the discrete logs of each factor base elements found in stage 2 be given by $b_i \equiv \log_g p_i$. Then a solution to the DLP is given by

$$\log_g h \equiv (\sum_{i=1}^{\#\mathcal{F}} e'_i b_i) - k' \pmod{r}.$$

Lemma 1.5.2. *Let $G = \mathbb{F}_p^*$ such that $G = \langle g \rangle$, where p is a large prime. Let $h \equiv g^a \pmod{p}$. The index calculus algorithm solves $\log_g h$ in subexponential time given by*

$$O\left(e^{c(\ln p)^{1/2}(\ln \ln p)^{1/2}}\right)$$

arithmetic group operations in G for some constant $c \in \mathbb{R}$ greater than zero.

Proof. See pages 301–334 of [Gal12] for sketch of the proof. □

Chapter 2

Elliptic Curves and Summation Polynomials

Contents

The main goal of this chapter is to explain the state of current research on solving the discrete logarithm problem using the index calculus algorithm in elliptic curves defined over a field extension $\mathbb{F}_{q^n}$ of characteristic greater than three. The main idea of this approach is to translate the ECDLP to the problem of solving systems of multivariate polynomial equations, and also linear algebra. We explain how symmetries coming from low order points of curves are used to speed up the index calculus algorithm using summation polynomials and Weil descent of elliptic curves.

We explore some existing algorithms to solve the elliptic curve discrete logarithm problem. We also give an overview of some of the mathematical concepts needed for this chapter. Some of the concepts from algebraic geometry we need to use include ideals and varieties, Gröbner basis, invariant theory and polynomial system solving.

2.1 Computational algebraic geometry

In this section we introduce some concepts from algebraic geometry such as ideals, varieties and Gröbner basis which are important for our applications. Our goal is to solve the elliptic curve discrete logarithm problem. We will come to understand that solving the elliptic curve discrete logarithm problem is transferred to solving a system of multivariate polynomial equations. Gröbner basis algorithms such as the F4 and F5 are currently the most effective algorithms to solve the latter. The main complexity indicator of these algorithms is the degree of regularity. We address the degree of regularity as our analysis critically depends on this value.

Our system of multivariate polynomial equations has high degree. However, there is a natural group action on it. Invariant theory provides a mechanism to lower the degree. This speeds up the algorithms to find solutions. So we give an elementary introduction to invariant theory. We also show how to achieve the degree reduction.

2.1.1 Ideals and affine varieties

Notation: We assume $\mathbb{K}$ is a field and $\overline{\mathbb{K}}$ is its algebraic closure. We use $x_1, x_2, \cdots, x_n$ as indeterminate variables in monomial and polynomial expressions. A good reference can be found in [CLO07].

Definition 2.1.1. *Let $(\alpha_1, \alpha_2, \cdots, \alpha_n) \in \mathbb{Z}^n_{\geq 0}$. A monomial m in the variables $x_1, x_2, \cdots, x_n$ is a product of the form*

$$x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}.$$

The monomial m can be abbreviated by x^α, where

$$\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$$

is the vector of exponents of the monomial m. The sum of the vector exponents constitutes the total degree of m. This is denoted as $\deg(m) = |x^\alpha| = \sum_{i=1}^{n} \alpha_i$.

Definition 2.1.2. *A polynomial f in the variables $x_1, x_2, \cdots, x_n$ with coefficients in $\mathbb{K}$ is a finite linear combination of monomials which can be written in the form*

$$\sum_\alpha b_\alpha x^\alpha \text{ for some } b_\alpha \in \mathbb{K}.$$

The set of all polynomials in the variables $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbb{K}$ is denoted as $\mathbb{K}[x_1, x_2, \cdots, x_n]$. In the expression of the polynomial f, b_α is called the coefficient of the monomial x^α. If $b_\alpha \neq 0$, then $b_\alpha x^\alpha$ is called a term. The total degree of f, denoted as $\deg(f)$, is given by

$$\deg(f) = \max\{|\alpha| \mid b_\alpha \neq 0\}.$$

A polynomial f is said to be homogenous if all its monomials with non-zero coefficients have the same total degree.

Definition 2.1.3. *Let $\mathcal{I} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]$ be a non-empty subset, then $\mathcal{I}$ is called an ideal if it satisfies the following properties:*

1. *If $f, g \in \mathcal{I}$ then $f + g \in \mathcal{I}$,*
2. *If $f \in \mathcal{I}$ then for all $h \in \mathbb{K}[x_1, x_2, \cdots, x_n]$, $hf \in \mathcal{I}$.*

Let $f_1, f_2, \cdots, f_k \in \mathbb{K}[x_1, x_2, \cdots, x_n]$, define $\langle f_1, f_2, \cdots, f_k\rangle$ to be

$$\langle f_1, f_2, \cdots, f_k\rangle = \left\{\sum_{i=1}^{k} h_i f_i \mid h_i \in \mathbb{K}[x_1, x_2, \cdots, x_n]\right\}.$$

Then $\langle f_1, f_2, \cdots, f_k\rangle$ is the ideal generated by $f_1, f_2, \cdots, f_k$.

Theorem 2.1.4. *(Hilbert's Theorem) An ideal $\mathcal{I} \in \mathbb{K}[x_1, x_2, \cdots, x_n]$ is finitely generated, that is, $\mathcal{I} = \langle f_1, f_2, \cdots, f_k\rangle$ for some k.*

Definition 2.1.5. *Let $\mathcal{F} = \{f_1, f_2, \cdots, f_k\} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]$, then the set*

$$V(\mathcal{F}) = \{(a_1, a_2, \cdots, a_n) \in \overline{\mathbb{K}}^n \ \mid\ \forall\ f \in \mathcal{F},\ \ f(a_1, a_2, \cdots, a_n) = 0\}$$

is called an affine variety.

Definition 2.1.6. *Let $V \subset \mathbb{K}^n$ be an affine variety, then the set*

$$\mathcal{I}(V) = \{f \in \mathbb{K}[x_1, x_2, \cdots, x_n] \ \mid\ f(a_1, a_2, \cdots, a_n) = 0\ \ \forall\ (a_1, a_2, \cdots, a_n) \in V\}$$

is an ideal and it is called the ideal of V.

Let $\mathcal{I}_1 = \langle f_1, f_2, \cdots, f_k\rangle$, then we observe that $\mathcal{I}_1 \subset \mathcal{I}(V(\mathcal{I}_1))$. In general we have the following proposition.

Proposition 2.1.7. *Let $V, W \subset \mathbb{K}^n$ be affine varieties. Then*

(i) $V \subset W \iff \mathcal{I}(V) \supset \mathcal{I}(W)$.

(ii) $V = W \iff \mathcal{I}(V) = \mathcal{I}(W)$.

Definition 2.1.8. *Let $\mathcal{I} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]$ be an ideal. Then the set*

$$\sqrt{\mathcal{I}} = \{g \in \mathbb{K}[x_1, x_2, \cdots, x_n] \ \mid\ g^m \in \mathcal{I} \ \text{for some}\ \ m \geq 1\}$$

is called the radical of the ideal $\mathcal{I}$. An ideal $\mathcal{I}$ is said to be a radical ideal if $\sqrt{\mathcal{I}} = \mathcal{I}$.

Definition 2.1.9. *Let $\mathcal{I} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]$ be an ideal. If $\#V(\mathcal{I}) < \infty$, then the dimension of the ideal $\mathcal{I}$ is zero.*

Definition 2.1.10. *Let $\mathcal{I} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]$ be an ideal of dimension zero. Then the dimension as a $\mathbb{K}-$vector space of $\mathbb{K}[x_1, x_2, \cdots, x_n]/\mathcal{I}$ is called the degree of $\mathcal{I}$. We refer to pages 232–236 of [CLO07] for details.*

Proposition 2.1.11. *$\mathcal{I} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]$ be an ideal of dimension zero with degree D. Then $\#V(\mathcal{I}) \leq D$ with equality if $\mathcal{I}$ is radical. In other words, the degree of $\mathcal{I}$ is equal to the number of solutions of $\mathcal{I}$ in $\overline{\mathbb{K}}$ counted with multiplicities.*

Proof. See pages 234–236 of [CLO07]. □

2.1.2 Gröbner basis

An ideal has lots of possible sets of generators. Some choices of generators make it easier to understand the ideal and its properties, and make computational problems easier to solve. A Gröbner basis of an ideal of multivariate polynomials is one such set of generators. A Gröbner basis depends on a choice of monomial ordering, so that is the first concept that needs to be defined. In particular, a Gröbner basis with respect to the lexicographic monomial order can be used to efficiently find points in the algebraic set defined by the ideal.

Definition 2.1.12. *A relation $>$ on $\mathbb{Z}^n_{\geq 0}$ is a monomial ordering if it satisfies the following three properties.*

1. *$>$ is a total ordering on $\mathbb{Z}^n_{\geq 0}$.*
2. *If $\alpha > \beta$ then $\alpha + \gamma > \beta + \gamma$, for all $\{\alpha, \beta, \gamma\} \subset \mathbb{Z}^n_{\geq 0}$.*
3. *$>$ is a well-ordering.*

Such a relation defines an ordering on $\{x^\alpha | \alpha \in \mathbb{Z}^n_{\geq 0}\} \subseteq \mathbb{K}[x_1, x_2, \cdots, x_n]$.

The lexicographic and degree reverse lexicographic monomial orderings are important for our application and they are defined as follows.

Definition 2.1.13. *Let $x^\alpha, x^\beta \in \mathbb{K}[x_1, x_2, \cdots, x_n]$ be monomials. We define the lexicographic ordering $>_{lex}$ by $x^\alpha >_{lex} x^\beta$ if and only if the left most non-zero entry of $\alpha - \beta$ is positive.*

Definition 2.1.14. *Let $x^\alpha, x^\beta \in \mathbb{K}[x_1, x_2, \cdots, x_n]$ be monomials. We define the graded reverse lexicographic ordering $>_{grlex}$ by $x^\alpha >_{grlex} x^\beta$ if and only if $|x^\alpha| > |x^\beta|$ or $|x^\alpha| = |x^\beta|$ and $x^\alpha >_{lex} x^\beta$.*

We are now in a position to define a Gröbner basis with respect to either of these monomial orderings. Define the leading term of the polynomial $f = \sum_\alpha b_\alpha x^\alpha$ to be

$$\mathrm{LT}(f) = \max_> \{b_\alpha x^\alpha \ \mid \ b_\alpha \neq 0\},$$

where $\max_>$ is with respect to the monomial ordering $>$. The Gröbner basis is defined as follows.

Definition 2.1.15. *Let $\mathcal{I} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]$ be an ideal. Let $\mathrm{LT}(\mathcal{I})$ be the set of leading terms of $\mathcal{I}$ with respect to some fixed ordering. Denote the ideal generated by the elements of $\mathrm{LT}(\mathcal{I})$ to be $\langle \mathrm{LT}(\mathcal{I}) \rangle$. Then the set $\{g_1, g_2, \cdots, g_t\} \subset \mathcal{I}$ is a Gröbner basis of the ideal $\mathcal{I}$ if*

$$\langle \mathrm{LT}(g_1), \mathrm{LT}(g_2), \cdots, \mathrm{LT}(g_t) \rangle = \langle \mathrm{LT}(\mathcal{I}) \rangle.$$

The set $g = \{g_1, g_2, \cdots, g_t\} \subset \mathcal{I}$ is a Gröbner basis if the leading term of every non-zero element in $\mathcal{I}$ is divisible by some leading term of g. By Hilbert's theorem, one can show that a Gröbner basis always exists for an ideal $\mathcal{I}$ and $\mathcal{I} = \langle g_1, g_2, \cdots, g_t \rangle$, which implies $V(\mathcal{I}) = V(g)$.

Given an ideal $\mathcal{I} = \langle f_1, f_2, \cdots, f_s \rangle$, we can compute its Gröbner basis using the Buchberger [Buc06], F4 [Fau99] and F5 [Fau02] algorithms. The Buchberger algorithm idea is to extend the generating set for $\mathcal{I}$ to a Gröbner basis by adding new polynomials in $\mathcal{I}$. The algorithm iteratively cancels the leading terms of $\mathcal{I}$ and introduces possible new ones, by considering all pairs in the generating set, using the so called Buchberger's criterion. To define the Buchberger's criterion we first define the $\mathrm{S-polynomial}$ of two polynomials f and g.

Definition 2.1.16. *Let $f, g \in \mathbb{K}[x_1, x_2, \cdots, x_n]$ be non-zero polynomials. Let* $\mathrm{LT}(f) = ax^\alpha$ and $\mathrm{LT}(g) = bx^\beta$ for some $a, b \in \mathbb{K}$. Assume x^γ is the least common multiple of x^α and x^β, $x^\gamma = \mathrm{lcm}(x^\alpha, x^\beta)$. Then the $\mathrm{S-polynomial}$ of f and g, written as $S(f,g)$, is the polynomial obtained by a linear combination of f and g given as

$$S(f,g) = \frac{x^\gamma}{\mathrm{LT}(f)} \cdot f - \frac{x^\gamma}{\mathrm{LT}(g)} \cdot g.$$

We observe that $S(f,g) \in \langle f, g \rangle$. Let $G = \{g_1, g_2, \cdots, g_t\} \subset \mathcal{I}$. Define $\overline{S(g_i, g_j)}^G$, for all pairs $i \neq j$, to be the remainder of $S(f,g)$ on division by G. Then Buchberger's criterion states that G is a Gröbner basis if and only if $\overline{S(g_i, g_j)}^G = 0$, for all pairs $i \neq j$.

The F4 [Fau99] and F5 [Fau02] are more recent algorithms for computing a Gröbner basis of a given ideal $\mathcal{I}$. These algorithms involve performing linear algebra on the so called *Macaulay matrix* [Mac16, Mac27] as studied by Lazard [Laz83].

Definition 2.1.17. *(Macaulay matrix) Let $F = \{f_1, f_2, \cdots, f_l\} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]$ be a set of polynomials of degrees less than or equal to d. Consider the set of all monomials T of degree less than or equal to d sorted in descending order for a fixed monomial ordering. The Macaulay matrix M_d is the matrix obtained by multiplying each polynomial $f_i \in F$ by all monomials $t_j \in T$ such that $\deg(f_i) + \deg(t_j) \leq d$. The coefficients of the product $t_j f_i$ constitute the rows of the Macaulay matrix M_d and the columns correspond to the monomials.*

Applying Gaussian elimination on the Macaulay matrix M_d eliminates the largest monomial terms and produces new polynomials which are algebraic combinations of the original polynomials.

The F4 and F5 Gröbner basis algorithms successively construct a Macaulay matrix M_d of increasing size until a Gröbner basis is reached. The complexity of these two algorithms is determined by the cost of the Gaussian elimination. In particular the highest degree d reached during the creation of the Macaulay matrices M_d determines the cost of Gaussian elimination (See [BFS04]). The highest such degree d is called the degree of regularity and denoted as d_{reg}.

Theorem 2.1.18. *[FGHR13] Let d_{reg} be the degree of regularity and let n be the number of variables of a zero-dimensional system. Then the arithmetic complexity of the F4 and F5 Gröbner basis algorithms is given by*

$$O\left(\binom{n + d_{\mathrm{reg}} - 1}{d_{\mathrm{reg}}}^\omega\right)$$

field operations, where $\omega < 3$ is the linear algebra constant.

The number of monomials determine the size of M_d with the largest size given by $\binom{n+d_{\mathrm{reg}}-1}{d_{\mathrm{reg}}}$. For large n compared with d_{reg}, the number of monomials is approximated to $n^{d_{\mathrm{reg}}}$. The time and memory complexity of computing a Gröbner basis using the F4 or F5 algorithms is then approximated to $n^{\omega d_{\mathrm{reg}}}$ and $n^{2d_{\mathrm{reg}}}$ respectively.

The F4 and F5 Gröbner basis algorithms are optimized to work with graded reverse lexicographic ordering. Change of ordering algorithms such as FGLM [FGLM93] and Gröbner walk [CKM97] are used to convert a Gröbner basis of a given order to other orderings. For our application, the FGLM algorithm is used to change a Gröbner basis in graded reverse lexicographic ordering into a Gröbner basis in lexicographic ordering. The lexicographic ordering is suitable for recovering solutions of zero dimensional polynomial systems (see Section 2.1.4).

Theorem 2.1.19. *[FGLM93] Let G be a Gröbner basis with respect to graded reverse lexicographic ordering in n variables of a zero-dimensional system. Let D be the degree of the ideal generated by G,* *equivalently the number of solutions counted with multiplicities over the algebraic closure of* $\mathbb{K}$*. Then the complexity of computing a Gröbner basis* $\tilde{G}$ *with respect to lexicographic ordering is given by*

$$O\left(nD^3\right)$$

field operations.

2.1.3 Invariant theory

In our applications, we will have a finite group acting on our system of multivariate polynomial equations. These systems of multivariate polynomial equations are invariant under the action of the finite group. We need to compute the generators of the invariant ring under the group in consideration. Thus we study invariant theory. The motivation is that re-writing our system of equations in terms of invariants reduces the complexity of solving it.

Definition 2.1.20. *Let* $GL(n,\mathbb{K})$ *be the set of all invertible* $n \times n$ *matrices with entries in* $\mathbb{K}$*. Let* $G \subset GL(n,\mathbb{K})$ *be a finite group. Suppose* G *acts on* $\mathbb{K}^n$ *by the usual matrix multiplication. Let* $f \in \mathbb{K}[x_1, x_2, \cdots, x_n]$ *and* $\sigma \in G$*. Then we define* $\sigma.f \in \mathbb{K}[x_1, x_2, \cdots, x_n]$ *by*

$$(\sigma.f)(\mathbf{y}) = f(\sigma^{-1}\mathbf{y}) \quad \textit{for all} \quad \mathbf{y} = (x_1, x_2, \cdots, x_n) \in \mathbb{K}^n.$$

This defines an action of G *on* $\mathbb{K}[x_1, x_2, \cdots, x_n]$*. Moreover a polynomial* $f \in \mathbb{K}[x_1, x_2, \cdots, x_n]$ *is invariant under* G *if for all* $\sigma \in G$*, it holds* $\sigma.f = f$*. The set of all invariant polynomials under* G *is given by*

$$\mathbb{K}[x_1, x_2, \cdots, x_n]^G = \left\{ f \in \mathbb{K}[x_1, x_2, \cdots, x_n] \mid \sigma.f = f \quad \textit{for all} \quad \sigma \in G \right\}.$$

By Hilbert's theorem, the invariant ring $\mathbb{K}[x_1, x_2, \cdots, x_n]^G$ is finitely generated. Its Hironaka decomposition (see Chapter 2 of [DK02]) is given by

$$\mathbb{K}[x_1, x_2, \ldots, x_n]^G = \bigoplus_{i=1}^{t} \eta_i \mathbb{K}[\theta_1, \theta_2, \cdots, \theta_n],$$

where $\theta_1, \theta_2, \cdots, \theta_n, \eta_1, \eta_2, \cdots, \eta_t \in \mathbb{K}[x_1, x_2, \cdots, x_n]^G$. We call $\{\theta_1, \theta_2, \cdots, \theta_n\}$ an algebraically independent set of primary invariants and the set $\{\eta_1, \eta_2, \cdots, \eta_t\}$ is called an algebraically dependent set of secondary invariants.

Definition 2.1.21. *Let* $\mathcal{H} \subset \mathbb{K}^n$ *be a hyperplane. Then a pseudo-reflection is a linear automorphism of* $\mathbb{K}^n$ *that is not the identity map but leaves* $\mathcal{H}$ *point-wise invariant. A group* $G \subset GL(n,\mathbb{K})$ *is a pseudo-reflection group if it is generated by pseudo-reflections.*

If G is a pseudo-reflection group [Kan01], the secondary invariants are reduced to 1 (see [Che55, ST54]). Thus the invariant ring of G is a polynomial algebra,

$$\mathbb{K}[x_1, x_2, \cdots, x_n]^G = \mathbb{K}[\theta_1, \theta_2, \cdots, \theta_n].$$

This gives an isomorphism between $\mathbb{K}[x_1, x_2, \cdots, x_n]^G$ and $\mathbb{K}[y_1, y_2, \cdots, y_n]$ in the indeterminate variables y_i. Under this isomorphism, a polynomial $f \in \mathbb{K}[y_1, y_2, \cdots, y_n]$ is mapped to $f(\theta_1, \theta_2, \cdots, \theta_n) \in \mathbb{K}[x_1, x_2, \cdots, x_n]^G$.

Invariant theory deals with finding the generators of the invariant ring $\mathbb{K}[x_1, x_2, \cdots, x_n]^G$ under the group G. If the characteristic of $\mathbb{K}$ does not divide the order of G, then averaging over the whole group G using the Reynolds operator is a suitable procedure for finding the generators of the invariant ring $\mathbb{K}[x_1, x_2, \cdots, x_n]^G$.

Definition 2.1.22. *Let $G \subset GL(n, \mathbb{K})$ be a finite group. The Reynolds operator of G is the map $R_G : \mathbb{K}[x_1, x_2, \cdots, x_n] \to \mathbb{K}[x_1, x_2, \cdots, x_n]$ given by*

$$R_G(f)(\mathbf{x}) = \frac{1}{|G|} \sum_{\sigma \in G} f(\sigma.\mathbf{x}).$$

Consider the symmetric group S_n. A polynomial $f \in \mathbb{K}[x_1, x_2, \cdots, x_n]$ is symmetric if it is invariant under the permutation of its variables. Let f be symmetric then $f \in \mathbb{K}[x_1, x_2, \cdots, x_n]^{S_n}$. The invariant ring of S_n is generated by the symmetric invariant polynomials

$$\begin{array}{rcl} e_1 & = & x_1 + x_2 + \cdots + x_n, \\ e_2 & = & x_1x_2 + x_1x_3 + \cdots x_{n-1}x_n, \\ \vdots & & \\ e_n & = & x_1x_2x_3 \cdots x_n. \end{array} \tag{2.1}$$

So we have $\mathbb{K}[x_1, x_2, \cdots, x_n]^{S_n} = \mathbb{K}[e_1, e_2, \cdots, e_n]$. We observe that the symmetric group S_n is a pseudo-reflection group (see Definition 2.1.21).

Apart from the invariant rings under the symmetric group S_n, we will consider invariant rings under the dihedral coxeter group D_n having order $2^{n-1}n!$ given by the semi-direct product

$$D_n = (\mathbb{Z}/2\mathbb{Z})^{n-1} \rtimes S_n.$$

In [FGHR13], the action of D_n on a polynomial $f \in \mathbb{K}[x_1, x_2, \cdots, x_n]^{D_n}$ arises as a permutation and an even number of sign changes in the variables due to the actions of S_n and $(Z/2Z)^{n-1}$ respectively. The invariant ring of D_n (as shown in [FGHR13]) is given by

$$\mathbb{K}[x_1, x_2, \cdots, x_n]^{D_n} = \mathbb{K}[p_2, p_4, \cdots p_{2(n-1)}, e_n] \quad \text{or} \quad \mathbb{K}[x_1, x_2, \cdots, x_n]^{D_n} = \mathbb{K}[s_1, s_2, \cdots, s_{n-1}, e_n],$$

where

$$p_i = \sum_{k=1}^{n} x_k^i \ , \quad s_i = \sum_{1 \le k_1 < \cdots k_i \le n} \prod_{j=1}^{i} x_{k_j}^2, \quad \text{and} \quad e_n = x_1 x_2 \cdots x_n. \tag{2.2}$$

Note that the s_i, p_i, e_n are the i^{th} elementary symmetric polynomials in the variables $x_1^2, x_2^2, \cdots, x_n^2$, the i^{th} power sum and the n^{th} elementary symmetric polynomial in the variables $x_1, x_2, \cdots, x_n$ respectively. The secondary invariants are reduced to 1. Thus we observe the dihedral coxeter group D_n is a pseudo-reflection group (see Definition 2.1.21).

The generators of the invariant ring under the groups S_n and D_n are used in [Gau09] and [FGHR13] respectively to lower the cost of polynomial system solving. The action D_n on a polynomial $f \in \mathbb{K}[x_1, x_2, \cdots, x_n]^{D_n}$, where the characteristic of $\mathbb{K}$ is 2, will arise naturally in our applications (see Section 3.2.3). More precisely the action of S_n arises as a permutation of variables and the action of $(Z/2Z)^{n-1}$ is from an even number of additions of 1 in the variables.

In general when the characteristic of $\mathbb{K}$ divides the order G, finding the generators of the invariant ring $\mathbb{K}[x_1, x_2, \cdots, x_n]^G$ is cumbersome. For the case when the characteristic of $\mathbb{K}$ is 3, we will adopt an *ad hoc* technique to find the generators of an invariant ring under G using the Singular computer algebra system [DGPS15].

2.1.4 Solving polynomial systems with symmetries using Gröbner basis

Definition 2.1.23. *Let $\mathcal{F}$ be a zero dimensional system given by a set of polynomial equations*

$$\mathcal{F} = \{f_1, f_2, \cdots, f_m\}, \quad \textit{where} \quad f_i \in \mathbb{K}[x_1, x_2, \cdots, x_n].$$

Then finding the zeroes of $\mathcal{F}$, $(z_1, z_3, \cdots, z_n) \in \overline{\mathbb{K}}^n$ such that

$$\begin{cases} f_1(z_1, z_2, \cdots, z_m) & = 0 \\ f_2(z_1, z_2, \cdots, z_m) & = 0 \\ & \vdots \\ f_m(z_1, z_2, \cdots, z_m) & = 0 \end{cases}$$

is called the polynomial systems solving problem.

We solve a given polynomial system using the F4 or F5 Gröbner basis algorithm. Computing Gröbner basis in degree reverse lexicographic ordering is fast compared to computing Gröbner basis in lexicographic ordering since the latter involves elimination of variables and the resulting Gröbner basis set is relatively larger than the former. On the other hand, Gröbner basis in lexicographic ordering is suitable for retrieving the solutions of polynomial systems. Indeed, Gröbner basis in lexicographic ordering produces a triangular form

$$\begin{cases} h_{1,1}(x_1, x_2, \cdots, x_n), & \cdots, h_{1,j_1}(x_1, x_2, \cdots, x_n), \\ h_{2,1}(x_2, \cdots, x_n), & \cdots, h_{2,j_2}(x_2, \cdots, x_n), \\ \vdots & \\ h_{n-1,1}(x_{n-1}, x_n), & \cdots, h_{n-1,j_1}(x_{n-1}, x_n), \\ h_n(x_n), & \end{cases}$$

for some j_i, where $1 \leq i \leq n-1$. The solutions can be read off from this triangular form by first factoring the univariate polynomial $h_n(x_n)$ using the Berlekamp algorithm [GG02] and then getting the values of x_n. Then by back substitution, the entire solution can be recovered.

So the solving strategy includes two steps. First a Gröbner basis in degree reverse ordering is computed using the F4 or F5 algorithms (see Theorem 2.1.18) and then a lexicographic ordering Gröbner basis is computed using the FGLM (see Theorem 2.1.19) change of ordering algorithm whose complexity depends on the number of solutions.

Theorem 2.1.24. *(Bezout's theorem [Wal78]) Let $\mathcal{F} = \{f_1, f_2, \cdots, f_m\} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]$ be a zero dimensional polynomial system such that* $\deg(f_i) = d_i$ for $1 \leq i \leq m$. Then $\mathcal{F}$ has at most $\prod_{i=1}^{m} d_i$ solutions in the algebraic closure $\overline{\mathbb{K}}$ counting with multiplicities.

If the number of solutions to the system of equations is small (for example when the system is overdetermined), the cost of solving the polynomial system is dictated by the complexity of the F4 or F5 algorithms. In fact the change of ordering FGLM algorithm is not needed. The Gröbner basis obtained in both orderings will be the same.

To effectively determine the complexity of F4 or F5, an accurate estimate of the degree of regularity (see Section 2.1.2) is needed. The degree of regularity of random polynomial system [Bar04] $\mathcal{F} = \{f_1, f_2, \cdots, f_m\} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]$ is given by

$$1 + \sum_{i=1}^{m} (\deg(f_i) - 1). \tag{2.3}$$

The exact estimate of the degree of regularity of polynomial systems having some structure (overdetermined systems, sparse systems, systems having symmetries) is an open problem. Experimental evidence [PQ12] on binary systems shows that for most systems the first fall degree (see [FJ03, DG10]), a degree fall that occurs during Gröbner basis computation, is approximately equal to the degree of regularity. Thus the degree of regularity can be approximated by the first fall degree.

Definition 2.1.25. *Let $\mathcal{F} = \{f_1, f_2, \cdots, f_m\} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]$ be a polynomial system. The first fall degree of $\mathcal{F}$ is the smallest degree d_{ff} such that there exists polynomials $g_i \in \mathbb{K}[x_1, x_2, \cdots, x_n]$ with the property $\max_i(\deg(g_i) + \deg(f_i)) = d_{ff}$ satisfying $0 < \deg(\sum_{i=1}^{m} g_i f_i) < d_{ff}$.*

The approximation of the degree of regularity by the first fall degree can be explained by observing how a Gröbner basis is obtained by the F4 and F5 algorithms. Note that the F4 and F5 Gröbner basis algorithms first fix a maximal degree d occurring in our system of polynomial equations to create a Macaulay matrix M_d (see Definition 2.1.17). Then, an initial Gaussian elimination is applied on M_d. If new low degree polynomials are obtained, they will be multiplied by all monomials and they are added to M_d. A Gaussian elimination is again applied on M_d; and the process is repeated until Gröbner basis is reached . If no new low degree polynomials are obtained, the maximal degree d is increased to allow new equations of low degree polynomials to be added through the Gaussian elimination process and the iteration continues until Gröbner basis is reached. Thus when a degree fall occurs during the Gröbner basis computation, the maximal degree d can be approximated by the first fall degree.

We now explain why symmetries can be used to reduce the complexity of polynomial system solving. Let $\mathcal{F} = \{f_1, f_2, \cdots, f_m\} \subset \mathbb{K}[x_1, x_2, \cdots, x_n]^G$, where G is a finite group. Then the ideal $\mathcal{I} = \langle \mathcal{F} \rangle$ is invariant under the action of G. The variety $V(\mathcal{F}) = V(\mathcal{I})$ is also invariant under the action of G on $\mathbb{K}^n$. Define the orbit space of the group G as follows.

Definition 2.1.26. *(See [FR09]) Let $\mathbf{a} = (a_1, \cdots, a_n) \in \mathbb{K}^n$ be a point. The orbit of $\mathbf{a}$ under the group G is the set $\{\sigma.\mathbf{a} \mid \sigma \in G\}$ and it is called the $G-$orbit of $\mathbf{a}$. The set of all $G-$orbits of $\mathbb{K}^n$, denoted as $\mathbb{K}^n/G$, is called the orbit space of G.*

Definition 2.1.27. *Let $\mathcal{I} = \langle \mathcal{F} \rangle$. Then the relative orbit variety, denoted as $V(\mathcal{I})/G$, is the set of points whose $G-$orbits are zeroes of $\mathcal{I}$.*

The polynomial system $\mathcal{F}$ admits a polynomial change of variables using the generators of the invariant ring $\mathbb{K}[x_1, x_2, \cdots, x_n]^G$ [FGHR13]. Write $\mathcal{F}$ using the primary invariants $\theta_1, \cdots, \theta_n \in \mathbb{K}[x_1, x_2, \cdots, x_n]^G$. So for each polynomial $f_i \in \mathcal{F}$, we have $f_i = g_i(\theta_1, \theta_2, \cdots, \theta_n)$ for some $g_1, g_2, \cdots, g_m \in \mathbb{K}[y_1, y_2, \cdots, y_n]$.

Instead of solving $\mathcal{F}$ directly, we solve the system $\mathcal{G} = \{g_1, g_2, \cdots, g_m\}$. In other words, we compute a Gröbner basis of the ideal of $\mathcal{G}$ having variety $V(\mathcal{I})/G$ instead of the original ideal $\mathcal{I}$ having variety $V(\mathcal{I})$. We now explain how to reconstruct $V(\mathcal{I})$ from $V(\mathcal{I})/G$. Let $\tilde{\mathbf{y}} = (\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, \cdots, \tilde{\mathbf{y}}_n) \in V(\mathcal{I})/G$, then a solution to the system of equations

$$\{\theta_1(x_1, \cdots, x_n) - \tilde{\mathbf{y}}_1, \theta_2(x_1, \cdots, x_n) - \tilde{\mathbf{y}}_2, \cdots, \theta_n(x_1, \cdots, x_n) - \tilde{\mathbf{y}}_n\}$$

gives a point in $V(\mathcal{I})$ which is in the orbit $\tilde{\mathbf{y}}$.

From the discussion above we observe that the solutions of the polynomial system $\mathcal{G}$ are the orbits of $V(\mathcal{I})$ under the action of G. The number of solutions of the ideal of $\mathcal{G}$ is less by a factor of $|G|$ than the number of solutions of the ideal generated by the original system $\mathcal{F}$ (equivalently the degrees of the ideals). Thus a reduction in the complexity of the FGLM (see Theorem 2.1.19) by a factor of $(|G|)^3$.

We use this idea to speed up the index calculus algorithm using symmetries coming from low order points of elliptic curves to solve polynomial system of equations obtained by applying Weil descent on summation polynomials of elliptic curves.

Another approach to solve polynomial systems defined over fields of characteristic 2 is to use SAT solvers. We compare Gröbner basis and SAT solvers to solve polynomial systems defined over $\mathbb{F}_2$ in Chapter 3 Section 3.5.

2.2 Elliptic curves

In this section we formally state the elliptic curve discrete logarithm problem. We recall the generic algorithms to solve the elliptic curve discrete logarithm problem in exponential time. We also give emphasis that the discrete logarithm problem is easy for certain types of curves which can be avoided by selecting appropriate elliptic curve parameters.

We know that the discrete logarithm problem in the case of finite fields can solved in subexponential time using the index calculus algorithm. The main aim of our research is to have a similar subexponential time algorithm for solving the elliptic curve discrete logarithm problem. We will show that the elliptic curve discrete logarithm problem is transferred to solving a system of multivariate polynomial equations. In this section we study elliptic curve representations such as twisted Edwards curves and binary Edwards curves. The reason for studying these curve equations is that addition by points of order 2 and 4 are easily expressed in terms of the curve coordinates. Hence we can consider larger group actions on the corresponding system of multivariate polynomial equations which ultimately speeds up their resolution of solving them.

2.2.1 Elliptic curve definition

Definition 2.2.1. *Let $\mathbb{K}$ be a field. An elliptic curve over $\mathbb{K}$ is defined by a non-singular affine Weierstrass equation*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{2.4}$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$.

By non-singular we mean the curve is smooth: $\frac{\partial E}{\partial x}$ and $\frac{\partial E}{\partial y}$ do not vanish simultaneously. In other words the system of equations

$$\begin{cases} y^2 + a_1xy + a_3y & = x^3 + a_2x^2 + a_4x + a_6 \\ a_1y - 3x^2 - 2a_2x - a_4 & = 0 \\ 2y + a_1x + a_3 & = 0 \end{cases}$$

has no common solutions in $\overline{\mathbb{K}}$.

By applying a change of coordinates, the elliptic curve E can be transformed into a short Weierstrass form.

Theorem 2.2.2. *Let E be the elliptic curve given in (2.4). Assume $\tilde{E}$ is another elliptic curve given by*

$$\tilde{E} : y^2 + \tilde{a}_1xy + \tilde{a}_3y = x^3 + \tilde{a}_2x^2 + \tilde{a}_4x + \tilde{a}_6.$$

Then E and $\tilde{E}$ are said to be isomorphic over $\mathbb{K}$ if there exists $u, r, s, t \in \mathbb{K}$ such that the change of coordinates

$$(x, y) \mapsto (u^2x + r, u^3y + u^2sx + t)$$

transforms the curve E to $\tilde{E}$.

Proof. See Chapter 3 Section 3 of [Sil09]. □

Corollary 2.2.3. *(Short Weierstrass form) Assume* $\operatorname{char}(\mathbb{K}) \notin \{2, 3\}$. Let E be an elliptic curve given by the Weierstrass equation as in (2.4). Then there exist $a, b \in \mathbb{K}$ such that the elliptic curve $\tilde{E}$ given by

$$\tilde{E} : y^2 = x^3 + ax + b$$

is isomorphic to E.

Proof. Let $\tilde{y} = y + (a_1x + a_3)/2$. Since char$(\mathbb{K}) \neq 2$, completing the square method on the left side of the curve equation (2.4) gives

$$\tilde{y} = x^3 + \frac{d_2}{4}x^2 + \frac{d_4}{2}x + \frac{d_6}{4}, \tag{2.5}$$

where $d_2 = a_1^2 + 4a_2, d_4 = 2a_4 + a_1a_3$ and $d_6 = a_3^2 + 4a_6$. As char$(\mathbb{K}) \neq 3$, substituting x by $\tilde{x} = x + \frac{d_2}{12}$ on the right hand side of equation (2.5) gives

$$\tilde{y} = \tilde{x}^3 - \frac{c_4}{48}\tilde{x} - \frac{c_6}{864},$$

where $c_4 = d_2^2 - 24d_4$ and $c_6 = -d_2^3 + 36d_2d_4 - 216d_6$. The transformation

$$(x, y) \mapsto \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{a_1^3 + 4a_1a_2 - 12a_3}{24}\right),$$

taking the values of $(u, r, s, t) = (\frac{1}{6}, -\frac{a_1^2+4a_2}{12}, -\frac{a_1}{2}, -\frac{a_1^3+4a_1a_2-12a_3}{24})$, gives the required isomorphism from E to $\tilde{E}$ for the pair $(a, b) = (-\frac{c_4}{48}, -\frac{c_6}{864})$. □

If char$(\mathbb{K}) = 3$, applying a change of coordinate to the elliptic curve E given in equation (2.4) gives the curves

$$y^2 = x^3 + a_2x^2 + a_6 \ \ (a_2, a_6 \neq 0) \text{ and } \ \ y^2 = x^3 + a_4x + a_6 \ (a_4 \neq 0),$$

where the latter is a supersingular curve (see Definition 2.2.10). Similarly if char$(\mathbb{K}) = 2$, applying a change of coordinates gives two families of curves

$$y^2 + xy = x^3 + a_2x^2 + a_6 \ \ (a_6 \neq 0) \text{ and } \ \ y^2 + a_3y = x^3 + a_4x + a_6 \ (a_3 \neq 0),$$

where the latter is a supersingular curve. For more details, we refer to Chapter 3 Section 1 of [HVM04].

The set of points of an elliptic curve form an abelian group. The group law is constructed geometrically by the chord-and-tangent rule which has nice geometrical interpretations. To define the group law first we introduce the projective curve.

Definition 2.2.4. *Let $\sim$ be an equivalence relation given by $(x_0, x_1, \cdots, x_n) \sim (y_0, y_1, \cdots, y_n)$ if there exists $\beta \in \mathbb{K}^*$such that $x_i = \beta y_i$ for $0 \leq i \leq n$. Then the projective n-space over $\mathbb{K}$, denoted by $\mathbb{P}^n$, is defined to be the set of all $(n + 1)$-tuples $(x_0, x_1, \cdots, x_n) \in \mathbb{K}^{n+1}$ such that $(x_0, x_1, \cdots, x_n) \neq (0, 0, \cdots, 0)$ modulo the equivalence relation $\sim$.*

We denote an equivalence class $\{(\beta x_0, \beta x_1, \cdots, \beta x_n)\}$ by $[x_0 : x_1 : \cdots : x_n]$. A polynomial $f \in \mathbb{K}[x_0, x_1, \cdots, x_n]$ is homogeneous of degree d if $f(\beta x_0, \cdots, \beta x_n) = \beta^d f(\beta x_0, \cdots, \beta x_n)$ for all $\beta \in \mathbb{K}^*$.

Definition 2.2.5. *Let $\mathcal{F} = \{f_1, f_2, \cdots, f_m\} \subseteq \mathbb{K}[x_0, x_1, \cdots, x_n]$ be homogenous polynomials. Then the set*

$$V(\mathcal{F}) = \{(a_0 : a_1 : \cdots : a_n) \in \mathbb{P}^n \quad | \quad f_i(a_0, a_1, \cdots, a_n) = 0 \quad \forall \quad 1 \leq i \leq m\},$$

is called a projective variety.

In fact an elliptic curve is a projective curve which is reduced to an affine curve by setting the third coordinate z to 1. Let $E : y^2 = x^3 + ax + b$ be a short Weierstrass form over $\mathbb{K}$, where char$(\mathbb{K}) \notin \{2, 3\}$. The corresponding homogenized curve equation of E is the projective curve given by

$$\tilde{E} : y^2z = x^3 + axz^2 + bz^3.$$

Observe that if $(x : y : z) \sim (x/z : y/z : 1) \in \tilde{E}$ then $(x/z, y/z) \in E$ for $z \neq 0$ and $(x, y, z) \neq (0, 0, 0)$. If $z = 0$ then $x = 0$. We define the point $(0 : y : 0) \sim (0 : 1 : 0)$ to be the point at infinity and we denote it ∞.

The set of points of the elliptic curve $E \cup \{\infty\}$ is a group with group operation given by a chord-and-tangent rule. The point at infinity ∞ acts as the identity element of the group. We focus on the group law given algebraically. We refer to Chapter 3 Section 1 of [Sil09, HVM04] for the geometrical construction of the group law.

Definition 2.2.6. *(Group Law) To define the group law algebraically, let $P_1 = (x_1, y_1), P_2 = (x_2, y_2) \in E$.*

1. ***(Identity)**: $P_1 + \infty = \infty + P_1 = P_1$.*
2. ***(Inverse)**: $-P_1 = (x_1, -y_1)$*
3. ***(Point Addition)**: Assume $P_1 \neq \pm P_2$ and let m be the slope joining P_1 and P_2, $m = \frac{y_2 - y_1}{x_2 - x_1}$, then $P_1 + P_2 = (x_3, y_3)$, where*
$$x_3 = m^2 - x_1 - x_2 \quad \textit{and} \quad y_3 = m(x_1 - x_3) - y_1.$$
4. ***(Point Doubling)**: Let $P_1 \neq -P_1$ and $m = \frac{3x_1^2 + a}{2y_1}$ be the slope of the line through P_1 which is tangent to the curve E, then $P_1 + P_1 = (x_3, y_3)$, where*
$$x_3 = m^2 - 2x_1 \quad \textit{and} \quad y_3 = m(x_1 - x_3) - y_1.$$

The number of points of an elliptic curve

Elliptic curves defined over finite fields play a central role in defining public key based cryptographic schemes. These schemes involve selection of a suitable curve E and a base point P. For wise selection of the parameters, it is important to understand the group structure of an elliptic curve which is determined by the number of rational points of the curve.

Theorem 2.2.7. *(Hasse) Let q be a prime power p^r where p is prime and $r \in \mathbb{Z}_{\geq 1}$. Let $\mathbb{K} = \mathbb{F}_q$. Let E be an elliptic curve defined over $\mathbb{K}$. Denote $\#E(\mathbb{K})$ to be the number of rational points in $E(\mathbb{K})$. Then*
$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{K}) \leq q + 1 + 2\sqrt{q}.$$

Definition 2.2.8. *Denote $[n]P$ to be $\underbrace{P + \cdots + P}_{n \text{ times}}$ for an integer $n \in \mathbb{Z}_{>0}$. Let E be an elliptic curve over $\mathbb{K}$. The set*
$$E[n] = \{P \in E(\overline{\mathbb{K}}) \mid [n]P = \infty\}$$
is called a torsion group. Any element $R \in E[n]$ is called a torsion element.

Definition 2.2.9. *Let E be an elliptic curve over $\mathbb{F}_p$ for a prime p. Then E is said to be an anomalous curve if $\#E(\mathbb{F}_p) = p$.*

Definition 2.2.10. *Let E be an elliptic curve over $\mathbb{F}_{q^n}$ for q a power of a prime p. Then E is said to be supersingular if $\#E(\mathbb{F}_{q^n}) = q^n + 1 - t$ where $p \mid t$.*

2.2.2 Elliptic curve representation

Elliptic curves can be represented in many forms. Some popular elliptic curve models include: Montgomery curves [Mon87], Twisted Edwards curve [BBJ+08] which encompass Edwards curve [Edw07, BL07], Twisted Jacobi intersection curves [FNW10] which contain Jacobi intersection curves [LS01], Hessian curves [JQ01, GGX11], Huff curves [Huf48, JTV10] and their variants. Some of these curves offer more efficient computations (reducing the cost of point doubling and point addition) and some provide resistance to side channel attacks.

Twisted Edwards Curve

Twisted Edwards curve is a generalization of Edwards curves [Edw07, BL07] introduced by Bernstein et al. [BBJ+08]. These curves were suitable for attacking the discrete logarithm problem associated with elliptic curves in [FGHR13, FHJ+14]. Particularly the symmetries coming from points of order 2 and 4, as we will describe shortly, are exploited to attack the discrete logarithm problem for elliptic curves.

Let $\mathbb{K}$ be a field of characteristic not equal to 2. Let $(a, d) \in \mathbb{K}$. The twisted Edwards curve is defined by

$$E_{a,d} \; : \; ax^2 + y^2 = 1 + dx^2y^2.$$

Let $P_1 = (x_1, y_1), P_2 = (x_2, y_2) \in E_{a,d}$. The group law of the twisted Edwards curve is defined as follows.

1. **(Identity)**: The neutral element is $(0, 0)$.
2. **(Inverse)**: Let $P = (x, y) \in E_{a,d}$, then $-P = (-x, y)$.
3. **(Point Addition)**: $P_1 + P_2 = (x_3, y_3)$, where

$$(x_3, y_3) \quad = \quad \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2} \right).$$

The point $T_2 = (0, -1)$ is a point of order 2. For any point $P = (x, y) \neq (0, 0)$, we have $P + T_2 = (-x, -y)$. If the constant a is a square in $\mathbb{K}$, then $(a^{-1/2}, 0)$ and $(-a^{-1/2}, 0)$ are points of order 4.

Binary Edwards curve

Bernstein et al. [BLF08] introduced the binary Edwards curve. This curve has points of order 2 and 4. We exploit the symmetry coming from these points to solve the discrete logarithm problem for this curve. This is possible mainly because the addition by points of order 2 and 4 is simpler than when using the Weierstrass model as noted in [FGHR13, FHJ+14].

Let $\mathbb{K}$ be a field of characteristic 2. Let $d_1, d_2 \in \mathbb{K}$ such that $d_1 \neq 0$ and $d_2 \neq d_1^2 + d_1$. The binary Edwards curve is given by

$$E_{d_1,d_2} \; : \; d_1(x + y) + d_2(x^2 + y^2) = xy + xy(x + y) + x^2y^2 \tag{2.6}$$

which is symmetric in the variables x and y.

Definition 2.2.11. *To define the group law of binary Edwards curve E_{d_1,d_2}, let $P_1 = (x_1, y_1), P_2 = (x_2, y_2) \in E_{d_1,d_2}$.*

1. ***(Identity)**: The neutral element is the point $P_0 = (0, 0)$.*
2. ***(Inverse)**: Let $P = (x, y) \in E_{d_1,d_2}$ then $-P = (y, x)$.*

3. ***(Point Addition)***: $P_1 + P_2 = (x_3, y_3)$, *where*

$$\begin{aligned} x_3 &= \frac{d_1(x_1+x_2) + d_2(x_1+y_1)(x_2+y_2) + (x_1+x_1^2)(x_2(y_1+y_2+1)+y_1y_2)}{d_1 + (x_1+x_1^2)(x_2+y_2)} \\ y_3 &= \frac{d_1(y_1+y_2) + d_2(x_1+y_1)(x_2+y_2) + (y_1+y_1^2)(y_2(x_1+x_2+1)+x_1x_2)}{d_1 + (y_1+y_1^2)(x_2+y_2)}. \end{aligned}$$

We observe that the point $T_2 = (1, 1) \in E_{d_1,d_2}$ is a point of order 2. Under the group law $P + T_2 = (x + 1, y + 1)$ for any point $P = (x, y) \in E_{d_1,d_2}$. The addition by T_2 to any point $(x, y) \in E_{d_1,d_2}$ is simple, adding 1 to each coordinate, which is useful for our attack. We will also notice that if the constants d_1 and d_2 are equal, we will obtain a point of order 4 whose addition to any point is not complicated (See Chapter 3 Section 3.2.3).

2.2.3 The elliptic curve discrete logarithm problem (ECDLP)

Elliptic curves were introduced for cryptography by Koblitz [Kob87] and Miller [Mil86], these curves defined over finite fields have become a substitute in the definition of public key cryptosystems that are close analogs of existing schemes such as Diffie-Hellman Key exchange schemes [DH76, ANSI97] and digital signature algorithms [ANSI98, NIST94]. Their applications range to primality testing and integer factorization [Men93, Len87].

Elliptic curves are good choices for the design of cryptosystems mainly because they offer relatively small key sizes [LV01] and more efficient computations [BL13]. More importantly the ECDLP, which is the heart of the security of these cryptosystems, has no known subexponential attack in general.

Definition 2.2.12. *(ECDLP) Consider an elliptic curve E defined over a field $\mathbb{K}$. Let $P \in E$ be a point having order n and let $Q \in E$. The elliptic curve discrete logarithm problem is to find β, if it exists, such that $Q = [\beta]P$.*

The ECDLP is believed to be a hard computational problem for an appropriate size of parameters. However there are some known attacks which could be easily avoided. Most of the attacks transfer the ECDLP to some other group where the DLP is easy. For example, the Weil descent and the GHS attacks [GHS02, GS99] transfer the DLP for elliptic curves defined over binary extension fields to DLP for hyperelliptic curves, where subexponential algorithms to solve the discrete logarithm problem exist [JMS01].

In what follows we focus on attacks that transfer the ECDLP to the DLP over finite field extensions. We also highlight that most of the algorithms discussed in Section 1.5 are not able to solve the ECDLP in subexponential time. The hope is to develop an index calculus algorithm to solve the ECDLP similar to the case it solves the DLP over finite fields and their extensions in subexponential time.

Generic attacks

Le $P \in E$ be a point of order n. Our goal is to find an integer β such that $Q = [\beta]P$. Algorithms that are used to solve the DLP over finite fields and their extensions namely baby-step-giant step, Pollard rho/kangaroo, Pohlig-Hellman as discussed in Section 1.5 are also applicable to solve the ECDLP. These algorithms do not take advantage of the structure of the group type, they are called generic algorithms. But their running time is exponential in the input size.

Since the Pohlig-Hellman algorithm reduces the DLP to subgroups of prime order, taking n prime, the best algorithm to attack the ECDLP among these generic attacks is the Pollard rho/kangaroo algorithm and its variants. Recall that the Pollard rho algorithm first defines a sequence of points in $\langle P \rangle$

$$X_0, \quad X_{i+1} = f(X_i),$$

where $X_i \in \langle P \rangle$. It then defines a pseudo random function $f : \langle P \rangle \mapsto \langle P \rangle$ (a similar function as discussed in Section 1.5 except that the group operation is now addition). Such a sequence will be ultimately periodic. The algorithm then tries to find a match in the sequence, $X_i = X_j$ (for $i \neq j$). By birthday paradox we expect to find a collision after $\sqrt{\frac{n\pi}{2}}$ iterations. Then a solution to the ECDLP will be found with high probability.

The Pollard rho algorithm can be parallelised [OM99] which improves the running time. Assume we have M processors coordinated by a central server. The server instantiates the processors with a random starting point X_0 for each processor using the same function f. We expect collision per processor to occur after $\frac{1}{M}\sqrt{\frac{n\pi}{2}}$ steps (see alo Chapter 4 Section 1 of [HVM04]).

The running time of the Pollard rho algorithm can be further improved using equivalence classes [GLV99, WZ98]. Suppose $\psi : \langle P \rangle \mapsto \langle P \rangle$ be an automorphism of groups. Let ψ has order k, $\psi^k(R_1) = R_1$ for any $R_1 \in \langle P \rangle$. Define an equivalence class

$$[R_1] = \{R_1, \psi(R_1), \psi^2(R_1), \cdots, \psi^{k-1}(R_1)\}.$$

Let $\tilde{R} = \{\tilde{R}_1, \tilde{R}_2, \tilde{R}_3, \cdots\}$ be the set of representatives of the equivalence class. Instead of applying the pseudo random function f on $\langle P \rangle$, we apply it on a well-defined and unique representative of the equivalence class $\tilde{R}$.

Assume we know $\psi(P) = [\alpha]P$ for some $\alpha \in \mathbb{Z}/n\mathbb{Z}$, then given $X = [a_1]P + [b_1]Q$ for some known random integer values $a, b \in \mathbb{Z}/n\mathbb{Z}$, we can determine its representative class $\tilde{x} = [a_2]P + [b_2]Q$. Indeed if $\tilde{X} = \psi^i(X)$, then $a_2 = \alpha^i a_1 \pmod{n}$ and $b_2 = \alpha^i b_1 \pmod{n}$.

If the size of each equivalence class is k, then the search space is reduced from n to $\frac{n}{k}$. So the expected running time of the parallelized Pollard rho using equivalence class is $O(\frac{1}{M}\sqrt{\frac{n\pi}{2k}})$.

Anomalous attack

Anomalous attack [SA98, Sem98, Sma99] exploits the group structure of the elliptic curve to attack the ECDLP. Recall that given E over $\mathbb{F}_q$, $\#E(\mathbb{F}_q) = q + 1 - t$, where $|t| \leq 2\sqrt{q}$ and q is a prime power. We specialize to the case when $q = p$ where p is a prime.

The attacks on anomalous elliptic curves due to Satoh-Araki [SA98], Semaev [Sem98] and Smart [Sma99] are based on the idea of transferring the ECDLP to a weaker group . All these attacks have running time $O(\log p)$.

Consider the attack by Smart [Sma99]. Let $E(\mathbb{Q}_p) : y^2 = x^3 + a_4 x + a_6$ be an elliptic curve defined over the p-adic field $\mathbb{Q}_p$ (see Chapter XI Section 6 of [Sil09]). Define the reduction modulo p map

$$\phi\ : E(\mathbb{Q}_p) \mapsto \tilde{E}(\mathbb{F}_p).$$

The n^{th} subgroup of $E(\mathbb{Q}_p)$ is defined to be $E_n(\mathbb{Q}_p) = \{W \in E(\mathbb{Q}_p) \mid v_p(x(W)) \leq -2n\} \cup \{\infty\}$, where $v_p(x(W))$ is the p-adic valuation of the x-coordinate of the point W. Particularly, we are interested in the first three groups of $E_n(\mathbb{Q}_p)$.

1. The kernel of ϕ, $E_1(\mathbb{Q}_p) = \{W \in E(\mathbb{Q}_p) \mid \phi(W) = \tilde{\infty}\}$.

2. $E_0(\mathbb{Q}_p) = \{W \in E(\mathbb{Q}_p) \mid \phi(W) \in \tilde{E}(\mathbb{F}_p)\}$.

3. The group $E_2(\mathbb{Q}_p) = \{W \in E(\mathbb{Q}_p) \mid v_p(x(W)) \leq -4\} \cup \{\infty\}$.

Theorem 2.2.13. *Define the p-adic elliptic logarithm map $\psi_p : E_1(\mathbb{Q}_p) \mapsto p\mathbb{Z}_p$ which sends $P \mapsto -\frac{x(P)}{y(P)}$ (see chapters* IV and VII in [Sil09]). Then

$$E(\mathbb{Q}_p)/E_1(\mathbb{Q}_p) \simeq \tilde{E}(\mathbb{F}_p) \;\text{ and }\; E_0(\mathbb{Q}_p)/E_1(\mathbb{Q}_p) \simeq E_1(\mathbb{Q}_p)/E_2(\mathbb{Q}_p) \simeq \mathbb{F}_p.$$

Proof. We refer to chapters IV and VII of [Sil09]. $\square$

Using the isomorphism given in theorem (2.2.13), we are in a position to describe the attack. Assume $\tilde{E}(\mathbb{F}_p) : y^2 = x^3 + \tilde{a}_4 x + \tilde{a}_6$ is an anomalous curve defined over the field $\mathbb{F}_p$. Let $\tilde{P}, \tilde{Q} \in \tilde{E}(\mathbb{F}_p)$ be such that $\tilde{Q} = [\beta]\tilde{P}$. The problem is to find β given $\tilde{R}$ and $\tilde{P}$. The First step is to lift the points $\tilde{Q}$ and $\tilde{P}$ to points $Q, P \in E(\mathbb{Q}_p)$ respectively using Hensel's Lemma [Sil09]. Then we observe that

$$Q - [\beta]P = R \in E_1(\mathbb{Q}_p).$$

Multiplying both sides by p results $[p]Q - [\beta][p]P = [p]R \in E_2(\mathbb{Q}_p)$ (Note that $[p]Q, [p]P \in E_1(\mathbb{Q}_p)$ and for $R \in E_1(\mathbb{Q}_p)$, then $[p]R \in E_2(\mathbb{Q}_p)$). Applying the p-adic elliptic logarithm ψ_p, we get

$$\psi_p([p]Q) - \beta\psi_p([p]P) \in p^2\mathbb{Z}_p.$$

This implies $\psi_p([p]Q) \equiv \beta\psi_p([p]P) \pmod{p^2}$. Since if $(x, y) \in E_1(\mathbb{Q}_p)$, $\psi_p(x, y) \equiv -\frac{x}{y} \pmod{p^2}$, we have $\beta = \frac{\psi_p([p]Q)}{\psi_p([p]P)} \pmod{p}$ which solves the ECDLP.

The Menezes-Okamoto-Vanstone (MOV) attack

The MOV [MOV93] attack uses the Weil pairing to transfer the DLP for elliptic curves defined over $\mathbb{F}_q$ to the DLP for the finite field extension $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$. One can then solve the DLP using the subexponential index calculus algorithm. We recall the definition of the Weil pairing.

Definition 2.2.14. *Let $E(\mathbb{F}_q)$ be an elliptic curve and n be positive integer such that GCD$(n, q) = 1$. Then the Weil pairing is the map*

$$e_n : E[n] \times E[n] \mapsto \mu_n,$$

where $\mu_n \in \overline{\mathbb{F}}_q$ is the n^{th} root of unity with the following properties.

1. ***(bilinear)**: Let $P, P_1, P_2, Q, Q_1, Q_2 \in E[n]$, then*

$$e_n(P_1 + P_2, Q) = e_n(P_1, Q)e_n(P_2, Q) \quad \text{and } e_n(P, Q_1 + Q_2) = e_n(P, Q_1)e_n(P, Q_2).$$

2. ***(Nondegenerate)**: $e_n(P, Q) = 1$ for all $Q \in E[n]$ if and only if $P = \infty$.*

3. ***(Identity)**: $e_n(P, P) = 1$ for all $P \in E[n]$.*

4. ***(Alternating)**: $e_n(P, Q) = e_n(Q, P)^{-1}$ for all $P, Q \in E[n]$.*

Let E be an elliptic curve defined over the field $\mathbb{F}_q$ such that $P \in E(\mathbb{F}_q)$ has order n. The idea of the MOV attack is to use the Weil pairing to form a group homomorphism from the group generated by P and the group of n^{th} roots of unity in $\mathbb{F}_{q^k}$, where k is the embedding degree defined as follows.

Definition 2.2.15. *Let E be an elliptic curve defined over the field $\mathbb{F}_q$. Let $P \in E(\mathbb{F}_q)$ be a point of order n. Assume* GCD$(n, q) = 1$. Then the embedding degree of $\langle P \rangle$ is the smallest integer k such that $n \mid q^k - 1$.

Theorem 2.2.16. *Let E be an elliptic curve defined over the field $\mathbb{F}_q$ such that $E[n] \subseteq E(\mathbb{F}_{q^k})$, where k is the embedding degree with GCD$(n, q) = 1$. Let $P \in E[n]$ be of order n. Then there exists $Q \in E[n]$ such that $e_n(P, Q)$ is a primitive n^{th} root of unity.*

Proof. Let $Q \in E[n]$, then $e_n(P, Q)^n = e_n(P, [n]Q) = e_n(P, \infty) = 1$, which implies $e_n(P, Q) \in \mu_n$. There are n cosets of $\langle P \rangle$ within $E[n]$. If $P_1, P_2 \in E[n]$ are in the same coset then $e_n(P, P_1) = e_n(P, P_2)$. So as we vary Q among the representative cosets, $e_n(P, Q)$ varies among the elements of μ_n. $\square$

Thus if $Q \in E[n]$ is such that $e_n(P,Q)$ is a primitive n^{th} root of unity, then the map

$$\begin{aligned} \phi : \langle P \rangle &\mapsto \mu_n \\ R &\mapsto e_n(P,R) \end{aligned}$$

is a group isomorphism.

Let $E(\mathbb{F}_q)$ be an elliptic curve such that $\#\langle P \rangle = n$, where $\text{GCD}(n,q) = 1$. Let $Q = [\beta]P$. Recall that the ECDLP problem is to find β. To recover β, the MOV attack proceeds by first finding the embedding degree k. We then find a random element $R \in E[n]$ such that $\alpha = e_n(P,R)$ has order n. Compute $\gamma = e_n(Q,R)$. Then

$$\gamma = e_n(Q,R) = e_n([\beta]P,R) = e_n(P,R)^\beta = \alpha^\beta \implies \beta = \log_\alpha \gamma \quad \text{in } \mathbb{F}_{q^k}.$$

The Weil pairing is efficiently computed using Miller's algorithm (see Chapter XI Section 8 in [Mil86] and [Mil04]). However for most elliptic curves the embedding degree k is large, as large as $k \approx n$, so that the index calculus method for solving the DLP in $\mathbb{F}_{q^k}$ becomes infeasible. Indeed it is shown in [BK98] that the probability of an elliptic curve of prime order defined over a prime field being susceptible to this attack is negligible. However, supersingular curves have small embedding degree $k \in \{1,2,3,4,6\}$ and are susceptible to the MOV attack. The attack could be foiled by avoiding low degree embedding curves in cryptographic applications.

The Frey-Rück attack

The Frey-Rück attack [FR94] makes use of the Tate-Lichtenbaum pairing instead of the Weil pairing to transfer the ECDLP to solving the DLP in $\mathbb{F}_{q^k}$, where k is the embedding degree.

Definition 2.2.17. *Let $E(\mathbb{F}_q)$ be an elliptic curve and $P \in E(\mathbb{F}_q)$ be of order n with GCD$(n,q) = 1$. Let $nE(\mathbb{F}_q) = \{nQ \mid Q \in E(\mathbb{F}_q)\}$. Then the Tate pairing is given by the map*

$$t: \ E(\mathbb{F}_q)[n] \times E(\mathbb{F}_q)/nE(\mathbb{F}_q) \mapsto \mathbb{F}_q^*/(\mathbb{F}_q^*)^n.$$

One can observe that the quotient group $\mathbb{F}_q^*/(\mathbb{F}_q^*)^n$ is isomorphic to the roots of unity μ_n. The Tate pairing outputs a coset in this group. In the modified Tate pairing, the output is raised to the power $(q^k-1)/n$ so that it is an element of the multiplicative group $\mathbb{F}_{q^k}^*$. The rest of the attack is quite similar to the MOV attack.

2.3 Summation polynomials

The idea of the index calculus algorithm to solve the elliptic curve discrete logarithm problem uses summation polynomials and Weil descent of an elliptic curve. The Weil descent is applied on the summation polynomial to get a system of multivariate polynomial equations. We discuss these concepts in detail accompanied with a complexity analysis of solving the resulting system of equations.

We also study how the naturally arising group action on the resulting system of equations coming from suitable elliptic curves such as the twisted Edwards curve or the binary Edwards curve speeds up the resolutions of solving the multivariate polynomial equations.

2.3.1 Summation polynomials definition

Definition 2.3.1. *Let E be an elliptic curve in Weierstrass form over a field $\mathbb{K}$. We define the m^{th} summation polynomial to be*

$$f_m(x_1, x_2, \cdots, x_m) \in \mathbb{K}[x_1, x_2, \ldots, x_m]$$

such that for all $X_1, X_2, \dots, X_m \in \overline{\mathbb{K}}$,

$$f_m(X_1, X_2, \cdots, X_m) = 0$$

if and only if there exist $Y_1, Y_2, \dots, Y_m \in \overline{\mathbb{K}}$ such that $(X_i, Y_i) \in E(\overline{\mathbb{K}})$ for all $1 \leq i \leq m$ and $(X_1, Y_1) + (X_2, Y_2) + \cdots + (X_m, Y_m) = \infty$, where ∞ is the identity element.

Theorem 2.3.2. *(Semaev [Sem04]) Let $E : y^2 = x^3 + a_4 x + a_6$ be an elliptic curve over a field $\mathbb{K}$ of characteristic $\notin \{2, 3\}$. The summation polynomials for E are given as follows.*

$$\begin{aligned}
f_2(x_1, x_2) &= x_1 - x_2. \\
f_3(x_1, x_2, x_3) &= (x_1 - x_2)^2 x_3^2 - 2((x_1 + x_2)(x_1 x_2 + a_4) + 2a_6)x_3 \\
&\quad +((x_1 x_2 - a_4)^2 - 4a_6(x_1 x_2)). \\
f_m(x_1, \cdots, x_m) &= \mathit{Resultant}_x(f_{m-j}(x_1, \cdots, x_{m-j-1}, x), f_{j+2}(x_{m-j}, x_{m-j+1}, \cdots, x_m, x)) \\
&\quad \textit{for all } m, \textit{ and } j, \textit{ such that } m \geq 4, \textit{ and } 1 \leq j \leq m - 3,
\end{aligned}$$

where $\mathit{Resultant}_x(f, g)$ is the resultant of the two polynomials f and g with respect to x. For $m \geq 3$, the m^{th} summation polynomial f_m is an irreducible symmetric polynomial having degree 2^{m-2} in each of the variables.

Proof. See Semaev [Sem04]. We will prove how to construct the 3^{rd} summation polynomial in the case of binary Edwards curve when we prove Theorem 3.1.1. □

2.3.2 Weil descent of an elliptic curve

Before we define the Weil descent of an elliptic curve, we illustrate the Weil descent concept by an example. Let $\mathbb{L}$ be a field extension of $\mathbb{K}$ such that $[\mathbb{L} : \mathbb{K}] = d$. The Weil descent relates a t dimensional object over $\mathbb{L}$ to a td dimensional object over $\mathbb{K}$. Assume $\mathbb{K} = \mathbb{F}_q$ and $L = \mathbb{F}_{q^2} = \mathbb{K}(i)$, where $i^2 = -1$. Let $x, y \in \mathbb{L}$, then $x = x_0 + ix_1$ and $y = y_0 + iy_1$, where $x_0, x_1, y_0, y_1 \in \mathbb{K}$.

Let $V \subseteq \mathbb{A}^2(\mathbb{K})$ be the affine algebraic set defined by $x^2 + y^2 - 1 = 0$. Then $(x_0 + ix_1)^2 + (y_0 + iy_1)^2 - 1 = 0$ if and only if

$$\begin{cases} x_0^2 - x_1^2 + y_0^2 - y_1^2 - 1 = 0, \\ 2x_0 x_1 + 2y_0 y_1 = 0. \end{cases}$$

If $W \subset \mathbb{A}^2(\mathbb{K})$ is the algebraic set of this system of equations. Then W defines the Weil descent of V. There is a bijection from $V(\mathbb{L})$ to $W(\mathbb{K})$.

Let E be an elliptic curve defined over $\mathbb{F}_{q^n}$. Write $\mathbb{F}_{q^n}$ as $\mathbb{F}_q[\theta]/(f(\theta))$, where $f(\theta)$ is an irreducible polynomial of degree n over the base field $\mathbb{F}_q$ and $\theta \in \mathbb{F}_{q^n}$ is a root. The Weil descent W of the affine curve E is the algebraic set of $2n$-tuples of elements $(x_0, x_1, \cdots, x_{n-1}, y_0, y_1, \cdots, y_{n-1}) \in \mathbb{F}_q^{2n}$ such that $x = x_0 + x_1\theta + \cdots + x_{n-1}\theta^{n-1}$ and $y = y_0 + y_1\theta + \cdots + y_{n-1}\theta^{n-1}$, where $(x, y) \in E(\mathbb{F}_{q^n})$. Since $y_0, y_1, \cdots, y_{n-1}$ are algebraic over $x_0, x_1, \cdots, x_{n-1}$, the dimension of W is n. The map $\phi : E(\mathbb{F}_{q^n}) \mapsto W(\mathbb{F}_q) \cup \{\infty\}$ is a bijection map and the group law on $E(\mathbb{F}_{q^n})$ corresponds to a group law on $W(\mathbb{F}_q) \cup \{\infty\}$.

2.3.3 The index calculus algorithm

The index calculus algorithm solves the discrete logarithm problem in the case of finite fields and their extensions in subexponential time. Motivated by the existence of such an algorithm, we wish to adopt the index calculus concept to the elliptic curve setting so that we may have a subexponential algorithm to solve the ECDLP.

Let E be an elliptic curve defined over the field $\mathbb{F}_{q^n}$. Let $Q = [\beta]P$. The idea of the index calculus algorithm to solve the ECDLP includes three stages.

– *Factor base definition:* The first stage is to define a suitable factor base $\mathcal{F} = \{P_1, P_2, \cdots, P_N\} \subset E(\mathbb{F}_{q^n})$.

– *Relation gathering:* The second stage is to decompose a random element $R \in E(\mathbb{F}_{q^n})$ as a sum of n elements of the factor base $\mathcal{F}$.

– *Linear algebra:* After collecting at least $\#\mathcal{F}$ relations in stage 2, apply Gaussian elimination on the set of relations to recover a solution to the ECDLP.

For an elliptic curve $E(\mathbb{F}_{q^n})$, where $n > 1$, Gaudry [Gau09] defined the factor base $\mathcal{F}$ to be

$$\mathcal{F} = \{P = (x, y) \in E(\mathbb{F}_{q^n}) \mid x \in \mathbb{F}_q\}.$$

The number of points in the factor base is $\#\mathcal{F} = q + O(\sqrt{q})$ (see [Wei49, CF05]). As $\sqrt{q}$ is small compared to q, $\#\mathcal{F} \approx q$. Once the factor base is defined, the next stage in the index calculus algorithm is to decompose a random element of an elliptic curve as a sum of the factor base elements. This is the critical stage of the algorithm.

Definition 2.3.3. *(Point Decomposition Problem) Let $E(\mathbb{F}_{q^n})$ be an elliptic curve. Let $\mathcal{F}$ be the factor base defined as above. Given a random element $R \in E(\mathbb{F}_{q^n})$, the Point Decomposition Problem (PDP) is to write R as a sum of n elements of the factor base $\mathcal{F}$,*

$$R = P_1 + P_2 + \cdots + P_n,$$

where $P_i \in \mathcal{F}$.

Gaudry used summation polynomials and Weil descent of an elliptic curve to solve the point decomposition problem and ultimately to solve the ECDLP. Assume the order of P is r and the ECDLP is to solve β such that $Q = [\beta]P$. Generate two random integers $(a, b) \in \mathbb{Z}/rZ$, then $R = [a]P + [b]Q$ is a random element in $E(\mathbb{F}_{q^n})$. To solve the PDP for the point R as

$$R = \epsilon_1 P_1 + \epsilon_2 P_2 + \cdots + \epsilon_n P_n, \tag{2.7}$$

where $P_i \in \mathcal{F}$ and $\epsilon_i \in \{-1, 1\}$, then it is enough to solve the summation polynomial

$$f_{n+1}(x(P_1), x(P_2), \cdots, x(P_n), x(R)) = 0, \tag{2.8}$$

where $x(P_i)$ denotes the x-coordinate of the point P_i. Sometimes we drop the notation $x(P_i)$ and instead use x_i, and in place of $x(R)$, we use x_R. Since $P_i \in \mathcal{F}$ we restrict $x_i \in \mathbb{F}_q$.

Let $\mathbb{F}_{q^n} = \mathbb{F}_q[x]/f(x)$, where $f(x) \in \mathbb{F}_q[x]$ is an irreducible polynomial of degree n having $\theta \in \mathbb{F}_{q^n}$ as a root. Applying Weil descent to the summation polynomial, we obtain

$$f_{n+1}(x_1, x_2, \cdots, x_n, x_R) = 0 \iff \sum_{i=0}^{n-1} \phi_i(x_1, x_2, \cdots, x_n)\theta^i = 0,$$

where $\phi_i(x_1, x_2, \cdots, x_n) \in \mathbb{F}_q[x_1, x_2, \cdots, x_n]$. But notice that $\sum_{i=0}^{n-1} \phi_i(x_1, x_2, \cdots, x_n)\theta^i = 0$ if and only if the following system of polynomial equations have common zeroes.

$$\begin{cases} \phi_0(x_1, x_2, \cdots, x_n) = 0 \\ \phi_1(x_1, x_2, \cdots, x_n) = 0 \\ \vdots \\ \phi_{n-1}(x_1, x_2, \cdots, x_n) = 0 \end{cases} \tag{2.9}$$

Hypothesis 2.3.4. *(See [Gau09, FGHR13]) The system of polynomial equations (2.9) coming from the resolution of the PDP given by equation (2.7) are of dimension zero.*

Remark 2.3.5. *When we restrict x_i to be in $\mathbb{F}_q$ and add the field equations $x_i^q - x_i = 0$ to the system of polynomial equations (2.9), then we have a dimension zero. In all our discussions, the system of polynomial equations coming from the resolution of the PDP given in equation (2.7) are considered to be of dimension zero.*

So the system of polynomial equations (2.9) can be solved using the F4 or F5 Gröbner basis algorithms (see Theorem 2.1.18) for the graded reverse lexicographic ordering followed by the FGLM algorithm (see Theorem 2.1.19).

After collecting enough independent relations in the second stage of the index calculus algorithm, the third stage is to apply linear algebra. Basically, we build a matrix M such that the relations correspond to the rows and the factor base elements correspond to the columns of the matrix M. Assume we have collected $N > \#\mathcal{F}$ relations of the form

$$R_i = [a_i]P + [b_i]Q = \sum_{P_j \in \mathcal{F}} M_{i,P_j} P_j,$$

where the integer value $M_{i,P_j} \in [-n, n]$ corresponds to the entry of the i^{th} row and P_j column of the matrix M. Then there exists a kernel element $(v_1, v_2, \cdots, v_N) \neq 0$ such that $(v_1, v_2, \cdots, v_N)M = 0$. So for all $P_j \in \mathcal{F}$, $\sum_{i=1}^{N} v_i M_{i,P_j} = 0$. Then

$$\sum_{i=1}^{N} v_i R_i = \left(\sum_{i=1}^{N} v_i[a_i]\right)P + \left(\sum_{i=1}^{N} v_i[b_i]\right)Q = \sum_{P_j \in \mathcal{F}} \sum_{i=1}^{N} v_i M_{i,P_j} P_j = 0.$$

As $Q = [\beta]P$, $\left(\sum_{i=1}^{N} v_i[a_i]\right)P + \left(\sum_{i=1}^{N} v_i[b_i]\right)Q = \left(\sum_{i=1}^{N} v_i[a_i]\right)P + \left(\sum_{i=1}^{N} v_i[b_i]\right)[\beta]P = 0$. Thus

$$\beta = -\frac{\sum_{i=1}^{N} v_i a_i}{\sum_{i=1}^{N} v_i b_i} \pmod r,$$

is a solution to the ECDLP provided that $\sum_{i=1}^{N} v_i b_i$ is invertible mod r.

Variants of the index calculus attack

Gaudry noted that for an elliptic curve E over an extension field $\mathbb{F}_{q^n}$, with a suitable choice of factor base, the problem of finding solutions to summation polynomials can be approached using the Weil descent with respect to $\mathbb{F}_{q^n}/\mathbb{F}_q$. In other words, the problem of solving $f_{n+1}(x_1, \ldots, x_n, x(R)) = 0$ for $x_i \in \mathbb{F}_q$ can be reduced to a system of polynomial equations over $\mathbb{F}_q$ as shown by equation (2.9).

The cost of solving the system of polynomial equations depends on the number of variables and on the degree of the summation polynomials. The higher the number of variables and the degree of the summation polynomials, the higher the cost is. To resolve the system, Joux and Vitse [JV13] proposed an $n-1$ scheme. The idea is to decompose a random element as a sum of $n-1$ elements of the factor base $\mathcal{F}$ instead of as a sum of n factor base elements as in the Gaudry case. Accordingly we have

$$R = [a]P + [b]Q = P_1 + P_2 + \cdots + P_{n-1},$$

where $P_i \in \mathcal{F}$. So one needs to solve $f_n(x_1, x_2, \cdots, x_{n-1}, x_R) = 0$. The number of variables is reduced from n to $n-1$ and the total degree of the summation polynomial is reduced from 2^{n-1} to 2^{n-2} compared with solving the summation polynomial given by the equation (2.8). The Weil descent produces n polynomial equations in both cases. But with the new method we have an overdetermined system and the resolution is greatly sped up. The resolution of the system does not come for free. The probability of decomposing a random element is decreased. We cover the complexity in the following Section.

Gaudry's [Gau09] approach restricts x to be in the base field $\mathbb{F}_q$. So the factor base cannot be increased in this case. Diem's [Die11] approach to solving the ECDLP generalizes the factor base definition. Assume we want to decompose a random element as a sum of m factor base elements instead of n in the Gaudry case. Let ℓ be such that $n \approx \ell m$. Diem defined the factor base $\mathcal{F}$ to be

$$\mathcal{F} = \{(x, y) \in E(\mathbb{F}_q) \mid x_i \in V, \ \ y_i \in \mathbb{F}_{q^n}\},$$

where $V \subseteq \mathbb{F}_{q^n}$ is a vector space of dimension ℓ.

Complexity analysis

The total complexity of the index calculus algorithm is dominated by the complexity of the point decomposition problem and the linear algebra stage. First we give a complexity estimate of the PDP followed by the complexity of the linear algebra stage. A given point $R \in E(\mathbb{F}_{q^n})$ can be decomposed as a sum of n elements of the factor base $\mathcal{F}$, $\{(x, y) \in E(\mathbb{F}_{q^n}) \mid x \in \mathbb{F}_q, y \in \mathbb{F}_{q^n}\}$, with a probability approximately $\frac{1}{n!}$. Indeed consider the map

$$\begin{aligned} \psi : \mathcal{F}^n &\mapsto A, \\ (P_1, P_2, \cdots, P_n) &\mapsto P_1 + P_2 + \cdots + P_n. \end{aligned} \tag{2.10}$$

Then we have $\psi(\sigma(P)) = \psi(P)$ (see equation 3.4), where S_n denotes the symmetric group having order $n!$ and $\sigma \in S_n$, and so ψ is well-defined on $\mathcal{F}^n/S_n$. We use the approximations $\#A = q^n$ and $\#(\mathcal{F}^n/S_n) = q^n/n!$. The probability of a random element R to be decomposed as a sum of n elements of the factor base is given by

$$\frac{\#(\mathcal{F}^n/S_n)}{\#A} \approx \frac{1}{n!}.$$

Let $c(n, q)$ be the complexity of checking a given point R is actually decomposable in the factor base or not. This is simply the cost of solving the system of polynomial equations (2.9). Since we have to collect q relations, the total complexity of the relation search stage of the index calculus algorithm is $n!c(n, q)q$. From the definition of the factor base, $P \in \mathcal{F} \implies -P \in \mathcal{F}$. So by only keeping one representative (P or $-P$), the size of the factor base can be reduced to $q/2$ bringing the total cost of relation search stage to $n!c(n, q)q/2$.

We now estimate the cost of the linear algebra stage. The matrix M constructed from the relation search stage has roughly q rows and columns, with a maximum of n non-zero entries per row. Utilizing the sparse nature of the matrix M, the overall Gaussian elimination cost using Wiedemann algorithm [Wie86, CF05] is $O(nq^2 \log^2 q)$. So the total complexity of the index calculus algorithm is

$$O(n!c(n, q)q + nq^2 \log^2 q)$$

arithmetic operations.

Taking the Joux and Vitse [JV13] $n-1$ scheme, decomposing a random element as a sum of $n-1$ factor base elements, the probability of finding a relation is decreased from $\frac{1}{n!}$ to $\frac{1}{q(n-1)!}$. Indeed by the map described in equation (2.10), we have

$$\frac{\#(\mathcal{F}^{n-1}/S_{n-1})}{\#E} = \frac{\#\mathcal{F}^{n-1}}{(n-1)!\#E} \approx \frac{q^{n-1}}{(n-1)!q^n} = \frac{1}{q(n-1)!}.$$

The total complexity using this scheme is then given by

$$O((n-1)!c(n-1,q)q^2 + (n-1)q^2\log^2 q).$$

Clearly the cost of solving the system of equations is reduced from $c(n,q)$ to $c(n-1,q)$. The system has now $n-1$ variables and a total degree of 2^{n-2} in each variable and hence there is a speed up. Above all the system is now overdetermined and if a solution exists, we get only few solutions. The cost of $c(n-1,q)$ is determined by solving the system of polynomial equations using the F4 and F5 Gröbner basis algorithms (see Theorem 2.1.18). The change of ordering algorithm FGLM (see Theorem 2.1.19) is not needed. Indeed computing a Gröbner basis of an overdetermined system using the F4 or F5 algorithm in degree reverse lexicographic ordering produces the same result as computing in lexicographic ordering.

Theorem 2.3.6. *Let E be an elliptic curve defined over $\mathbb{F}_{q^n}$. For fixed n, Gaudry [Gau09] solves the ECDLP using index calculus algorithm in*

$$\tilde{O}(q^{2-2/n})$$

arithmetic operations, with a hidden constant $c(n,q)$, which is exponential in n.

Before we prove Theorem 2.3.6, we discuss a technique called 'double large prime variation' [GTTD07, Thé03], to balance the cost of the relation search stage and linear algebra stage. Assume the linear algebra cost is high relative to the relation search stage. According to [GTTD07, Thé03], if we reduce the size of the factor base, obviously the cost of the linear algebra is decreased. On the other hand, the cost of the relation search stage increases as the probability of decomposing a random element over the factor base is reduced.

To balance the two costs, the idea is to divide the factor base $\mathcal{F}$ into two sets $\mathcal{F}_1$ and $\mathcal{F}_2$. We call $\mathcal{F}_1$ a factor base and it contains $(\#\mathcal{F})^r \approx q^r$ elements, where $0 < r \leq 1$. Decomposition of a random element R of a curve over $\mathcal{F}_1$ corresponds to the normal decomposition

$$R = [a_i]P + [b_i]Q = P_1 + P_2 + \cdots + P_n, \tag{2.11}$$

where $P_i \in \mathcal{F}_1$. Where as the set $\mathcal{F}_2$ contains elements of the factor base $\mathcal{F}$ that are not considered in $\mathcal{F}_1$ and are called 'large primes'. Decomposition of a random element of R over the factor base $\mathcal{F}_1$ with respect to the set $\mathcal{F}_2$ has two forms

$$R = [a_i]P + [b_i]Q = P_1 + P_2 + \cdots + P_{n-1} + \tilde{P}_1, \tag{2.12}$$

$$R = [a_i]P + [b_i]Q = P_1 + P_2 + \cdots + P_{n-2} + \tilde{P}_1 + \tilde{P}_2, \tag{2.13}$$

where $\tilde{P}_1, \tilde{P}_2 \in \mathcal{F}_2$ and $P_1, \cdots, P_{n-1} \in \mathcal{F}_1$. Note that decomposition of the form (2.13) are relatively easier followed by (2.12) and (2.11).

We collect relations coming from these decompositions. Relations involving large primes are recorded using graphs (see [GTTD07, Thé03]). The idea is if during the collection phase, we encounter decomposition involving identical large primes either in (2.12), (2.13) or a combination of (2.12) and (2.13), then they can be combined together to form decomposition of the form (2.11). As a result, we obtain a new (possibly many) relation(s).

The size of the new factor base $\mathcal{F}_1$ is q^r, hence the complexity of the linear algebra is $\tilde{\mathcal{O}}(q^{2r})$. The complexity of the relation search stage is $\tilde{\mathcal{O}}\Big((1+r\frac{n-1}{n})(n-2)!q^{1+(n-2)(1-r)}c(n,q)\Big)$ (see [GTTD07]) arithmetic operations. The optimal value $r = 1 - \frac{1}{n}$ is obtained by setting $q^{2r} = q^{1+(n-2)(1-r)}$. Hence, the overall complexity is

$$\tilde{\mathcal{O}}(q^{2(1-\frac{1}{n})}) \tag{2.14}$$

arithmetic operations.

Proof. (Theorem 2.3.6) Let $E(\mathbb{F}_{q^n})$ be an elliptic curve. For fixed n, the cost of solving the system of polynomial equation (2.9) over $\mathbb{F}_q$ with fixed degree and number of variables is polynomial in $\log q$. So the overall cost of the index calculus algorithm is dominated by the linear algebra stage which is given by $\tilde{\mathcal{O}}(nq^2)$.

Gaudry used a technique called 'double large prime variation' [GTTD07, Thé03] to balance the cost of the relation search stage and linear algebra stage as discussed above. According to the large prime variation, the overall complexity is given by (2.14). Thus for fixed n, the cost of solving the index calculus algorithm using Gaudry's approach is $\tilde{\mathcal{O}}(q^{2-2/n})$. □

We can observe that Gaudry's approach to solve the ECDLP is faster than Pollard rho whose complexity is $O(q^{n/2})$ for $n \geq 3$. But the analysis hides the constant $c(n,q)$. Diem [Die13] also showed that if q is large enough then the ECDLP can be solved in an expected subexponential time.

To precisely estimate the cost of $c(n,q)$, recall that the usual strategy of solving the system of equation (2.9) as discussed in Section 2.1.4 is to first use the Gröbner basis algorithms F4 or F5 in degree reverse ordering and then a change of ordering FGLM algorithm is applied to get Gröbner basis in lexicographic ordering. So the complexity of the $c(n,q)$ involves the complexity of the two steps.

For the system of equations derived from the symmetric summation polynomial having n variables with a total degree of 2^{n-1} in each variable, Bezout Theorem 2.1.24 tells us that there are at most $n!2^{n(n-1)}$ solutions. The cost of recovering such solutions using the FGLM algorithm as described in Theorem 2.1.19 is

$$O\Big(nD^3\Big) = O\Big(n(n!2^{n(n-1)})^3\Big)$$

arithmetic operations in $\mathbb{F}_q$, where D is the number of solutions counted with multiplicities in the algebraic closure of $\mathbb{F}_q$.

The complexity of computing a Gröbner basis using the F4 or F5 algorithms as indicated in Theorem 2.1.18 is

$$O\left(\binom{n + d_{\text{reg}} - 1}{d_{\text{reg}}}^{\omega}\right)$$

field operations. This complexity can be upper bounded by

$$O\left(n\binom{n + d_{\text{reg}}}{n}^{\omega}\right)$$

arithmetic operations. But note that the F5 algorithm is more efficient than the F4 algorithm as its gets rid off useless computations.

As highlighted in Section 2.1.4, the precise estimate of the degree of regularity is not known for all types of system of polynomial equations. We make the following assumption as in [FGHR13].

Assumption 2.3.7. *Let E be an elliptic curve over $\mathbb{F}_{q^n}$, the degree of regularity of polynomial systems arising from the Weil descent (2.9) (see [BFSY05]) is estimated to be* $1 + \sum_{i=0}^{n-1}(\deg(\phi_i) - 1)$.

By Assumption 2.3.7, the degree of regularity d_{reg} of the system of polynomial equations (2.9), having n variables with degree 2^{n-1} in each variable is estimated to be

$$d_{\text{reg}} \leq 1 + \sum_{i=0}^{n-1} (\deg(\phi_i) - 1) \leq n2^{n-1} - n + 1.$$

Thus the complexity of computing a Gröbner basis using the F4 or F5 algorithms is given by

$$O\left(n\binom{n + d\text{reg}}{n}^{\omega}\right) \leq O\left(n\binom{n2^{n-1} + 1}{n}^{\omega}\right)$$

arithmetic operations. So the overall cost of $c(n, q)$ includes the complexity of computing a Gröbner basis using the F4 or F5 algorithms and the complexity of the change of ordering FGLM (see Theorem 2.1.19) algorithm and is given by

$$O\left(n\binom{n2^{n-1} + 1}{n}^{\omega} + n(n!2^{n(n-1)})^3\right) = O\left(n(n!2^{n(n-1)})^3\right). \tag{2.15}$$

We observe that the complexity of the FGLM algorithm is exponential in n and it is the main complexity indicator in the overall complexity analysis given in equation (2.15). If the number of solutions to the system of equations is reduced, we know the complexity of the FGLM will also be reduced. Previous and our research focus on minimizing the number of solutions of a system of polynomial equations using symmetries to lower the FGLM complexity.

However if we have an overdetermined system, such as polynomial systems obtained by Weil descent for the Joux and Vitse [JV13] $n - 1$ scheme (see also 3.3), the number of solutions is few. So the FGLM complexity is negligible and the complexity of solving an overdetermined system is determined by the complexity of the F4 or F5 Gröbner basis algorithms which in turn depends on the degree of regularity. The use of symmetries with these systems makes the polynomial equations compact and our experiment (see Section 3.6) shows that the degree of regularity is less compared to original system.

2.3.4 Resolution of polynomial systems using symmetries

The summation polynomials are symmetric. Gaudry [Gau09] noticed that these polynomials belong to the invariant ring under the symmetric group,

$$f_{n+1}(x_1, x_2, \cdots, x_n, x(R)) \in \mathbb{F}_{q^n}[x_1, x_2, \cdots, x_n]^{S_n},$$

where S_n is the symmetric group. The generators of the invariant ring $\mathbb{F}_{q^n}[x_1, x_2, \cdots, x_n]^{S_n}$ are the elementary symmetric polynomials $e_1, e_2, \cdots, e_m$. So the summation polynomial can be re-written in terms of $e_1, e_2, \cdots, e_n$ to get $\tilde{f}_{n+1}(e_1, e_2, \cdots, e_n, x_R) \in \mathbb{F}_{q^n}[e_1, e_2, \cdots, e_n]$. $\tilde{f}_{n+1}$ has n variables and a total degree of 2^{n-1}.

When we restrict $x_i \in \mathbb{F}_q$ then we also have $e_i \in \mathbb{F}_q$. Applying Weil descent to $\tilde{f}_{n+1}$, we get a system of equations denoted by $\mathcal{S} \subset \mathbb{F}_q[e_1, e_2, \cdots, e_n]$ with n polynomial equations and total degree 2^{n-1}. The degree of the ideal of the system $\mathcal{S}$ is bounded by $2^{n(n-1)}$. In other words we expect $2^{n(n-1)}$ solutions though most of them lie in a field extension of $\mathbb{F}_q$. By making use of the new variables, the degree of the ideal of the system $\mathcal{S}$ is less by $n!$ than the original system (equivalently the number of solutions are decreased by $n!$). This results a decrease in the FGLM cost by $(n!)^3$. See Section 2.1.4 for details.

In [FGHR13], the curve structure is exploited to further reduce the cost of polynomial system solving. Particularly, we consider the twisted Edwards curve and twisted Jacobi intersection curve instead of the Weierstrass model. Focusing on the former, that is on the twisted Edwards curve, the existence of low order rational points (order 2, and order 4 points) on these curves speeds up the resolution of polynomial systems arising from the Weil descent.

The twisted Edwards curve defined in Section 2.2 over $\mathbb{F}_{q^n}$ is given by $E_{a,d} \; : \; ax^2+y^2 = 1+dx^2y^2$, where $a, d \in \mathbb{F}_{q^n}$. We consider the point of order two $T_2 = (0, -1)$. For any point $P = (x, y) \in E_{a,d}$, by the group law $T_2 + P = (-x, -y)$.

The factor base $\mathcal{F}$ is defined in terms of the invariant variable under the map $[-1] : E_{a,d} \mapsto E_{a,d}$, $P \mapsto -P$. Note that $-P = (-x, y)$. So the y-coordinate remains invariant under the negation map. Accordingly the factor base is defined as

$$\mathcal{F} = \{(x, y) \in E_{a,d} \mid y \in \mathbb{F}_q, \;\; x \in \mathbb{F}_{q^n}\}.$$

The summation polynomials [FGHR13] are also constructed using the y-coordinate and are given by

$$\begin{aligned} f_2(y_1, y_2) &= y_1 - y_2, \\ f_3(y_1, y_2, y_3) &= (y_1^2y_2^2 - y_1^2 - y_2^2 + \frac{a}{d})y_3^2 + 2\frac{d-a}{d}y_1y_2y_3 \\ &\quad +\frac{a}{d}(y_1^2 + y_2^2 - 1) - y_1^2y_2^2, \\ f_n(y_1, \cdots, y_n) &= \textit{Resultant}_y(f_{n-j}(y_1, \cdots, y_{n-j-1}, y), f_{j+2}(y_{n-j}, y_{n-j+1}, \cdots, y_n, y)) \\ &\quad \textit{for all } n, \textit{ and } j, \textit{ such that } n \geq 4, \textit{ and } 1 \leq j \leq n-3 \;. \end{aligned}$$

For $n \geq 3$, the summation polynomial are symmetric, irreducible having n variables and a total degree of 2^{n-1} in each variable.

If $R = P_1 + P_2 + \cdots + P_n$ for $P_i \in \mathcal{F}$, then $f_{n+1}(y_1, y_2, \cdots, y_n, y(R)) = 0$, where $y_i = y(P_i)$ denotes the y-coordinate of the points P_i and $y(R)$ denotes a known y-coordinate value of R. Assume we have a decomposition $R = P_1 + P_2 + \cdots + P_n$ for $P_i \in \mathcal{F}$, then

$$R = (P_1 + u_1T_2) + (P_2 + u_2T_2) + \cdots + (P_n + u_nT_2),$$

where $(u_1, u_2, \cdots, u_n) \in \{0, 1\}$ such that $\displaystyle\sum_{i=1}^{n} u_i \pmod 2 = 0$. The observation is that an even number of additions of T_2 cancels out. From one decomposition of R, we get 2^{n-1} distinct decompositions for free by running over all possible values of $(u_1, u_2, \cdots, u_n)$. We also get 2^{n-1} distinct solutions corresponding to these decompositions. Indeed applying an even number of sign changes to the original solution $(y_1, y_2, \cdots, y_n)$ gives 2^{n-1} distinct solutions.

Consider the dihedral coxeter group $D_n = (\mathbb{Z}/2\mathbb{Z})^{n-1} \rtimes S_n$ having order $2^{n-1}n!$ acting on the solution set of the summation polynomial f_{n+1}, where $(\mathbb{Z}/2\mathbb{Z})^{n-1}$ changes the sign on an even number of the solutions and S_n permutes them, then the summation polynomials are invariant under the group action D_n. Thus

$$f_{n+1}(y_1, y_2, \cdots, y_n, y(R)) \in \mathbb{F}_{q^n}[y_1, y_2, \cdots, y_n]^{D_n},$$

where $\mathbb{F}_{q^n}[y_1, y_2, \cdots, y_n]^{D_n}$ is the invariant ring under the dihedral coxeter group D_n. This is a well known invariant ring generated either by the polynomials $(p_2, \cdots p_{2(n-1)}, e_n)$ or $(s_1, s_2, \cdots, s_{n-1}, e_n)$ (see equation (2.2)).

Writing the summation polynomial $f_{n+1}(y_1, y_2, \cdots, y_n, y(R))$ in terms of the new variables $(p_2, \cdots p_{2(n-1)}, e_n)$ or $(s_1, s_2, \cdots, s_{n-1}, e_n)$, we get $\tilde{f}_n(p_2, \cdots p_{2(n-1)}, e_n) \in \mathbb{F}_{q^n}[p_2, \cdots p_{2(n-1)}, e_n]$ or
$\tilde{f}_n(s_1, s_2, \cdots, s_{n-1}, e_n) \in \mathbb{F}_{q^n}[s_1, s_2, \cdots, s_{n-1}, e_n]$, where $\tilde{f}_n$ denotes f_{n+1} evaluated at $y(R)$. After Weil descent, we obtain a system of equations $\mathcal{S} \subset \mathbb{F}_q[p_2, \cdots p_{2(n-1)}, e_n]$ or $\mathcal{S} \subset \mathbb{F}_q[s_1, s_2, \cdots, s_{n-1}, e_n]$.

The degree of the ideal of $\mathcal{S}$ (see Proposition 2.1.11 and Definition 2.1.10) is then decreased by the order of D_n (see Section 2.1.4), in other words the number of solutions have decreased from $n!2^{n(n-1)}$ to $\frac{n!2^{n(n-1)}}{\#D_n} = \frac{n!2^{n(n-1)}}{n!2^{n-1}} = 2^{(n-1)^2}$. The overall effect is that we gain a speed up in the FGLM algorithm by $(n!2^{n-1})^3$.

Chapter 3

Index Calculus Algorithm to Solve the DLP for Binary Edwards Curve

Contents

This chapter presents our main results. We discuss the point decomposition problem to solve the DLP for Edwards curves defined over the field $\mathbb{F}_{2^n}$. With this curve, we get symmetries coming from the action of points of order 2 and 4. We exploit these symmetries to speed up the resolution of a system of polynomial equations over $\mathbb{F}_2$ obtained by applying Weil descent to the summation polynomials of this curve. We provide experimental evidence showing that our approach gives an improvement over previous work. We also give a comparison between using SAT solvers and using Gröbner basis methods for solving the system of polynomial equations.

Finally we describe some ideas to increase the probability of decomposing a random element of the curve as a sum of factor base elements with or without using symmetries coming from low order points. Following our work [GG14], a new idea based on "splitting" the summation polynomials is suggested. We discuss the splitting technique as our conclusion to this chapter.

This chapter is a joint work with Steven Galbraith.

3.1 Summation polynomials of binary Edwards curve

Recall from Section 2.2.2 that a binary Edwards curve is an affine curve given by equation (2.6),

$$E_{d_1,d_2}(\mathbb{F}_{2^n}) : d_1(x+y)+d_2(x^2+y^2) = xy + xy(x+y) + x^2y^2,$$

where $d_1 \neq 0$ and $d_2 \neq d_1^2 + d_1$ for $d_1, d_2 \in \mathbb{F}_{2^n}$. The conditions $d_1 \neq 0$ and $d_2 \neq d_1^2 + d_1$ ensure that the curve is non-singular. If $\mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_2}(d_2) = 1$, i.e., there is no element $v \in \mathbb{F}_{2^n}$ such that v satisfies $v^2 + v + d_2 = 0$, then the addition law on the binary Edwards curve is complete [BLF08]. In other words, the denominators in the addition law (see Definition 2.2.11), $d_1 + (y_1 + y_1^2)(x_2 + y_2)$ and $d_1 + (x_1 + x_1^2)(x_2 + y_2)$ never vanish.

We are interested in this curve because the point $T_2 = (1, 1)$ is a point of order 2 whose addition to any point in E_{d_1,d_2} is given by a simple formula. Such structures of curves defined over a field of characteristic greater than 3 have been exploited in [FGHR13] for solving the ECDLP.

We now define the summation polynomials for binary Edwards curve. For an elliptic curve E given by the Weierstrass equation, Semaev [Sem04] defined the summation polynomials in terms of the invariant variable x under the negation map $[-1] : E \mapsto E$, $P \mapsto [-]P$: If $P = (x, y) \in E$ then $[-]P = (x, -y)$. In [FGHR13], an elliptic curve $\tilde{E}$ represented by twisted Edwards curve (see Section 2.2) defined over a field of characteristic greater than 3 is considered. The summation polynomials for this curve are defined in terms of the invariant variable y under the negation map: If $P = (x, y) \in \tilde{E}$ then $[-]P = (-x, y)$.

In our case, we consider the function $t : E_{d_1,d_2}(\mathbb{F}_{2^n}) \to \mathbb{P}^1$, $t(P) = x(P) + y(P)$, where $x(P)$ and $y(P)$ denote the x-coordinate and y-coordinate of a point $P \in E_{d_1,d_2}$ respectively. This function is invariant under the action of the negation map. Note that if $P = (x, y) \in E_{d_1,d_2}(\mathbb{F}_{2^n})$, then $[-]P = (y, x)$. In [BLF08], the value $t(P)$ named as 'ω' is used for differential addition.

Theorem 3.1.1. *Let E_{d_1,d_2} be an Edwards curve over $\mathbb{F}_{2^n}$ with $P_0 = (0, 0)$ the identity element. Then the summation polynomials of E_{d_1,d_2} given by*

$$\begin{aligned}
f_2(t_1,t_2) &= t_1 + t_2 \\
f_3(t_1,t_2,t_3) &= (d_2t_1^2t_2^2 + d_1(t_1^2t_2 + t_1t_2^2 + t_1t_2 + d_1))t_3^2 + d_1(t_1^2t_2^2 + t_1^2t_2 + t_1t_2^2 + t_1t_2)t_3 \\
&\quad + d_1^2(t_1^2 + t_2^2) \\
f_m(t_1,\ldots,t_m) &= \mathit{Resultant}_u(f_{m-k}(t_1,t_2,\ldots,t_{m-k-1},u), f_{k+2}(t_{m-k},t_{m-k+1},\ldots,t_m,u)), \\
&\quad \text{for } m \geq 4 \text{ and } 1 \leq k \leq m-3,
\end{aligned}$$

have the following properties: for any points $P_1, \ldots, P_m \in E_{d_1,d_2}(\overline{\mathbb{F}_2})$ such that $P_1 + \cdots + P_m = P_0$, we have $f_m(t(P_1), \ldots, t(P_m)) = 0$. Conversely, given any $t_1, \ldots, t_m \in \overline{\mathbb{F}_2}$ such that $f_m(t_1, \ldots, t_m) = 0$, then there exist points $P_1, \ldots, P_m \in E_{d_1,d_2}(\overline{\mathbb{F}_2})$ such that $t(P_i) = t_i$ for all $1 \leq i \leq m$ and $P_1 + \cdots + P_m = P_0$. The summation polynomials are symmetric and irreducible having degree 2^{m-2} in each variable.

Proof. Due to the recursive construction of the summation polynomials, it is sufficient to prove the theorem for the case $m = 2$ and $m = 3$. Let $P_i = (x_i, y_i) \in E_{d_1,d_2}$ and $t_i = x_i + y_i$ for $1 \leq i \leq m$. We start with $m = 2$. If $P_1 + P_2 = P_0$ then $P_1 = -P_2 = (y_2, x_2)$ which implies $t(P_1) = y_2 + x_2 = t(P_2) = x_2 + y_2$. Thus $t_1 = t_2$ and it is clear to see that $f_2(t_1, t_2) = t_1 + t_2 = 0$. The other properties can also be easily verified.

For $m = 3$, we have to construct the 3^{rd} summation polynomial $f_3(t_1, t_2, t_3)$ corresponding to $P_1 + P_2 + P_3 = P_0$. We follow Semaev's [Sem04] construction method. Let $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$

and $(x_4, y_4) = (x_1, y_1) - (x_2, y_2)$. Applying the group law, we have

$$\begin{aligned} x_3 &= \frac{d_1(x_1+x_2)+d_2(x_1+y_1)(x_2+y_2)+(x_1+x_1^2)(x_2(y_1+y_2+1)+y_1y_2)}{d_1+(x_1+x_1^2)(x_2+y_2)}, \\ y_3 &= \frac{d_1(y_1+y_2)+d_2(x_1+y_1)(x_2+y_2)+(y_1+y_1^2)(y_2(x_1+x_2+1)+x_1x_2)}{d_1+(y_1+y_1^2)(x_2+y_2)}. \end{aligned}$$

So $t_3 = x_3 + y_3$ is given by

$$\begin{aligned} t_3 &= \frac{d_1(x_1+x_2)+d_2(x_1+y_1)(x_2+y_2)+(x_1+x_1^2)(x_2(y_1+y_2+1)+y_1y_2)}{d_1+(x_1+x_1^2)(x_2+y_2)} + \\ &\quad \frac{d_1(y_1+y_2)+d_2(x_1+y_1)(x_2+y_2)+(y_1+y_1^2)(y_2(x_1+x_2+1)+x_1x_2)}{d_1+(y_1+y_1^2)(x_2+y_2)}. \end{aligned}$$

Similarly we compute (x_4, y_4) to get

$$\begin{aligned} x_4 &= \frac{d_1(x_1+y_2)+d_2(x_1+y_1)(x_2+y_2)+(x_1+x_1^2)(y_2(y_1+x_2+1)+y_1x_2)}{d_1+(x_1+x_1^2)(x_2+y_2)}, \\ y_4 &= \frac{d_1(y_1+x_2)+d_2(x_1+y_1)(x_2+y_2)+(y_1+y_1^2)(x_2(x_1+y_2+1)+x_1y_2)}{d_1+(y_1+y_1^2)(x_2+y_2)}. \end{aligned}$$

Let $t_4 = x_4 + y_4$. Then we construct a quadratic polynomial in the indeterminate variable θ whose roots are t_3 and t_4, $\theta^2 + (t_3 + t_4)\theta + t_3t_4$. We can use the `EliminationIdeal()` function of Magma [BCP97] and the curve equation to express $t_3 + t_4$ and t_3t_4 in terms of the variables t_1 and t_2 to finally get

$$t_3 + t_4 = \frac{d_1t_1t_2(t_1t_2+t_1+t_2+1)}{d_1^2+d_1\left(t_1+t_1^2\right)t_2+\left(d_1t_1+d_2t_1^2\right)t_2^2} \quad \text{and}$$

$$t_3t_4 = \frac{d_1^2(t_1+t_2)^2}{d_1^2+d_1\left(t_1+t_1^2\right)t_2+\left(d_1t_1+d_2t_1^2\right)t_2^2}.$$

So the quadratic polynomial constructed is

$$\begin{aligned} \theta^2+(t_3+t_4)\theta+t_3t_4 &= \left(d_1^2+d_1\left(t_1+t_1^2\right)t_2+\left(d_1t_1+d_2t_1^2\right)t_2^2\right)\theta^2 \\ &\quad +\left(d_1t_1t_2(t_1t_2+t_1+t_2+1)\right)\theta+d_1^2(t_1+t_2)^2. \end{aligned}$$

Note that if $P_1 = P_2$ then $(x_3, y_3) = (x_1, y_1) + (x_1, y_1)$ then $t_3 = x_3 + y_3$ is also the root of the quadratic polynomial constructed above. So taking the summation polynomial

$$\begin{aligned} f_3(t_1,t_2,t_3) &= (d_2t_1^2t_2^2+d_1(t_1^2t_2+t_1t_2^2+t_1t_2+d_1))t_3^2+d_1(t_1^2t_2^2+t_1^2t_2+t_1t_2^2+t_1t_2)t_3+ \\ &\quad d_1^2(t_1+t_2)^2 \end{aligned}$$

gives the result. For $m \geq 4$ we use the fact that $P_1 + \cdots + P_m = P_0$ if and only if there exists a point R on the curve such that $P_1 + \cdots + P_{m-k-1} + R = P_0$ and $-R + P_{m-k} + \cdots + P_m = P_0$. It follows that

$$\begin{aligned} f_m(t_1,\ldots,t_m) &= \text{Resultant}_u(f_{m-k}(t_1,t_2,\ldots,t_{m-k-1},u), f_{k+2}(t_{m-k},t_{m-k+1},\ldots,t_m,u)), \\ &\quad (\text{for all } m \geq 4 \text{ and } m-3 \geq k \geq 1), \end{aligned}$$

where the resultant is taken with respect to the variable u.

We can observe that the 3^{rd} summation polynomial is symmetric from its construction and has degree 2 in each variable t_i. As shown by Semaev [Sem04], irreducibility follows from the fact that $f(t_1, t_2, t_3) = 0$ is isomorphic over $\overline{\mathbb{K}(x_3)}$ to the binary Edwards curve E_{d_1,d_2}. □

Corollary 3.1.2. *Let E_{d_1,d_2} be a binary Edwards curve. Let $d_1 = d_2$ and $P_0 = (0,0)$ be the identity point. Then the summation polynomials of E_{d_1,d_2} given by*

$$\begin{aligned} f_2(t_1,t_2) &= t_1+t_2, \\ f_3(t_1,t_2,t_3) &= (d_1+t_1t_2(t_1+1)(t_2+1))t_3^2+(t_1t_2+(t_1+1)(t_2+1))t_3+d_1(t_1+t_2)^2, \\ f_m(t_1,\ldots,t_m) &= \mathit{Resultant}_u(f_{m-j}(t_1,t_2,\ldots,t_{m-j-1},u), f_{j+2}(t_{m-j},t_{m-j+1},\ldots,t_m,u)) \\ & \quad (\textit{for all } m \geq 4 \textit{ and } 1 \leq j \leq m-3) \end{aligned}$$

satisfy the following properties: for any points $P_1,\ldots,P_m \in E_{d_1,d_2}(\overline{\mathbb{F}_2})$ such that $P_1+\cdots+P_m = P_0$, we have $f_m(t(P_1),\ldots,t(P_m)) = 0$. Conversely, given any $t_1,\ldots,t_m \in \overline{\mathbb{F}_2}$ such that $f_m(t_1,\ldots,t_m) = 0$, then there exist points $P_1,\ldots,P_m \in E_{d_1,d_2}(\overline{\mathbb{F}_2})$ such that $t(P_i) = t_i$ for all $1 \leq i \leq m$ and $P_1+\cdots+P_m = P_0$. The summation polynomials are symmetric and irreducible having degree 2^{m-2} in each variable.

Proof. Substitute the constant d_2 by d_1 for the 3^{rd} summation polynomial in Theorem 3.1.1 . Then the summation polynomial has d_1 as a factor. Since $d_1 \neq 0$, the result follows. □

We use the summation polynomials given by Corollary 3.1.2 through out this chapter.

3.1.1 Factor base definition

Definition 3.1.3. *Let $E_{d_1,d_2}(\mathbb{F}_{2^n})$ be a binary Edwards curve. Let $V \subset \mathbb{F}_{2^n}$ be a vector subspace of dimension ℓ. We define the factor base to be the set,*

$$\mathcal{F} = \{P \in E_{d_1,d_2}(\mathbb{F}_{2^n}) \mid t(P) \in V\}.$$

We heuristically assume that $\#\mathcal{F} \approx \#V$. Accordingly the size of the factor base is approximately 2^ℓ. Decomposing a random element $R \in E_{d_1,d_2}$ as a sum of m factor base elements will correspond to having relations of the form $R = P_1 + P_2 + \cdots + P_m$ where $P_i \in \mathcal{F}$. Hence, the probability that a uniformly chosen point $R \in E_{d_1,d_2}(\mathbb{F}_{2^n})$ can be decomposed as a sum of m factor base elements is

$$\frac{\#(\mathcal{F}^m/S_m)}{\#E_{d_1,d_2}} = \frac{\#\mathcal{F}^m}{m!\#E_{d_1,d_2}} \approx \frac{2^{\ell m}}{m!2^n} = \frac{1}{m!2^{n-\ell m}},$$

where S_m is the symmetric group of order $m!$. To have a solution on average we choose $\ell = \lfloor n/m \rfloor$, where $\lfloor n/m \rfloor$ denotes the nearest integer to n/m.

If $n = \ell m$, then V is a subfield of $\mathbb{F}_{2^n}$ and hence the factor base is defined over a subfield of $\mathbb{F}_{2^n}$. In this case, symmetries coming from the action of order 2 and 4 points reduce the complexity of the FGLM algorithm. We will discuss this briefly in Section 3.2.

If however n is prime, it is more difficult to get a good algorithm as the number of variables involved in the final system of equations becomes large. Nevertheless, we give experimental evidence (in Section 3.6) that using symmetries coming from these low order points produces a compact polynomial system which speeds up the resolution of polynomial systems obtained by applying Weil descent to the summation polynomials of the binary Edwards curve. The complexity of solving such polynomial systems is dictated by the complexity of the F4 or F5 Gröbner basis algorithms. Since the complexity of these algorithms is expressed in terms of the degree of regularity, we record the degree of regularity and compare our algorithm with previous works in our experimental Section 3.6.

3.1.2 Weil descent of binary Edwards curve

To decompose a random element R as a sum of m factor base elements, $R = P_1 + P_2 + \cdots + P_m$ where $P_i \in \mathcal{F}$, we need to solve the summation polynomial of the binary Edwards curve $f_{m+1}(t_1, t_2, \cdots, t_m, t_{m+1})$. If such a decomposition exists, then $f_{m+1}(t(P_1), \cdots, t(P_m), t(R)) = 0$.

First consider a special case where the vector subspace V is a subfield of $\mathbb{F}_{2^n}$ of dimension ℓ (in other words, $\ell | n$). Let $\mathbb{F}_{2^n} = \mathbb{F}_{2^\ell}[\theta]/f(\theta)$, where $f \in \mathbb{F}_{2^\ell}[\theta]$ is an irreducible polynomial of degree m having $\theta \in \mathbb{F}_{2^n}$ as a root. First we evaluate the summation polynomial $f_{m+1}(t_1, t_2, \cdots, t_m, t_{m+1})$ at t_{m+1} by substituting $t(R)$, then we apply Weil descent to obtain

$$\sum_{i=0}^{m-1} \phi_i(t_1, t_2, \cdots, t_m)\theta^i = 0.$$

This is true if and only if

$$\phi_0(t_1, t_2, \cdots, t_m) = \phi_1(t_1, t_2, \cdots, t_m) = \cdots = \phi_{m-1}(t_1, t_2, \cdots, t_m) = 0, \tag{3.1}$$

where $\phi_i(t_1, t_2, \cdots, t_m) \in \mathbb{F}_{2^\ell}[t_1, t_2, \cdots, t_m]$.

The polynomial system given by equation (3.1) has m equations and m variables. By Hypothesis 2.3.4, the polynomial system is of dimension zero. So we can solve this polynomial system using the F4 or F5 Gröbner basis algorithms (see Theorem 2.1.18) in degree reverse lexicographic ordering followed by the FGLM algorithm in lexicographic ordering. The complexity of solving such a polynomial system is governed by the complexity of the FGLM algorithm (see Theorem 2.1.19) given by $O(mD^3)$, where $D = m!2^{m(m-1)}$ is the degree of the ideal of the polynomial system. In this case, our goal is to reduce the complexity of the FGLM algorithm using symmetries coming from the action of points of order two and four.

If however n is prime and the factor base is defined with regard to the $\mathbb{F}_2$ vector subspace V of $\mathbb{F}_{2^n}$ of dimension ℓ such that $\ell = \lfloor n/m \rfloor$, then we get an overdetermined system of equations. Indeed consider $\mathbb{F}_{2^n}$ as a vector space over $\mathbb{F}_2$. Suppose $\mathbb{F}_{2^n}$ is represented using a polynomial basis $\{1, \theta, \ldots, \theta^{n-1}\}$ where $f(\theta) = 0$ for some irreducible polynomial $f(x) \in \mathbb{F}_2[x]$ of degree n. We will take V to be the vector subspace of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ with basis $\{1, \theta, \ldots, \theta^{\ell-1}\}$. Let $R = P_1 + P_2 + \cdots + P_m$ then $f_{m+1}(t(P_1), \cdots, t(P_m), t(R)) = 0$, where $t(P_j) \in V$ and $t(R) \in \mathbb{F}_{2^n}$. Write $t(R) = r_0 + r_1\theta + r_2\theta^2 + \cdots + r_{n-1}\theta^{n-1}$ with $r_i \in \mathbb{F}_2$, and $t(P_j)$ as

$$t(P_j) = \sum_{i=0}^{\ell-1} c_{j,i}\theta^i \tag{3.2}$$

where $1 \leq j \leq m$ and $c_{j,i} \in \mathbb{F}_2$. Let $\tilde{f}_m$ be f_{m+1} evaluated at $t(R)$. Then we have

$$\begin{aligned}
0 &= \tilde{f}_m(t_1, t_2, \cdots, t_m) \\
&= \tilde{f}_m(\sum_{i=0}^{\ell-1} c_{1,i}\theta^i, \sum_{i=0}^{l-1} c_{2,i}\theta^i, \cdots, \sum_{i=0}^{l-1} c_{m,i}\theta^i) \\
&= \sum_{i=0}^{n-1} \phi_i(c_{1,0}, c_{1,2}, \cdots, c_{1,\ell-1}, c_{2,0}, \cdots, c_{m,\ell-1})\theta^i \\
\iff & \phi_1(c_{1,0}, \cdots, c_{m,\ell-1}) = \phi_2(c_{1,0}, \cdots, c_{m,\ell-1}) = \cdots = \phi_{n-1}(c_{1,0}, \cdots, c_{m,\ell-1}) = 0
\end{aligned} \tag{3.3}$$

where $\phi_i \in \mathbb{F}_2[c_{1,0}, \cdots, c_{m,\ell-1}]$.

So applying Weil descent on $\tilde{f}_m$ gives a system of n equations in the ℓm binary variables $c_{j,i}$. As $n > \ell m$, we have an overdetermined system. Moreover adding the field equations $c_{j,i}^2 - c_{j,i} = 0$, we get a polynomial system with $n + \ell m$ equations and ℓm binary variables. It is clear that the number of solutions for an overdeterimined system is very small (actually in most cases it is zero). As a result the change of ordering algorithm FGLM (see Theorem 2.1.19) is not needed. So the complexity of solving the system of polynomial equations (3.3) is determined by the complexity of the F4 or F5 Gröbner basis algorithms. In Sections 3.3 and 3.6, we discuss how to solve this system of polynomial equations.

3.2 Symmetries to speed up resolution of polynomial systems

Let the polynomial system to be solved be the polynomial system of equations given by equation (3.1). We discuss how symmetries coming from the action of the symmetric group S_m, the action of points of order 2 and 4 speed up the complexity of solving this polynomial system.

3.2.1 The action of symmetric group

Let E_{d_1,d_2} be a binary Edwards curve. Let $P_i = (x_i, y_i) \in E_{d_1,d_2}$. Define the action of the symmetric group S_m on $(P_1, P_2, \cdots, P_m)$ by

$$\sigma(P_1, P_2, \cdots, P_m) = (P_{\sigma(1)}, P_{\sigma(2)}, \cdots, P_{\sigma(m)}), \tag{3.4}$$

where $\sigma \in S_m$. Then the symmetry in the addition of points $P_1 + \cdots + P_m = R$ induces a symmetry on the corresponding summation polynomials. As a result the summation polynomials of the binary Edwards curve E_{d_1,d_2} belong to the invariant ring under the symmetric group S_m. Let $\tilde{f}_m$ be f_{m+1} evaluated at $t(R)$ then

$$\tilde{f}_m(t_1, t_2, \ldots, t_m) \in \mathbb{F}_{2^n}[t_1, t_2, \ldots, t_m]^{S_m}.$$

We express the summation polynomial $\tilde{f}_m$ in terms of the elementary symmetric polynomials $e_1, e_2, \ldots, e_m$ to get $\tilde{f}_m(e_1, e_2, \cdots, e_m)$. As discussed in Section 2.3, the degree of the ideal of the polynomial system of equations obtained by applying Weil descent on $\tilde{f}_m(e_1, e_2, \cdots, e_m)$ is less by $m!$ than the polynomial system of equations given by equation (3.1). As a result the FGLM (see Theorem 2.1.19) complexity is reduced by a factor of $(m!)^3$.

3.2.2 The action of a point of order 2

As shown in Section 2.2, if $P = (x, y) \in E_{d_1,d_2}$ then $P + T_2 = (x + 1, y + 1)$. Note that $t(P + T_2) = (x + 1) + (y + 1) = x + y = t(P)$ so the function t is already invariant under addition by T_2. Since the factor base is defined in terms of $t(P)$ we have that $P \in \mathcal{F}$ implies $P + T_2 \in \mathcal{F}$.

Let $R = P_1 + \cdots + P_m$ and let $u = (u_1, \ldots, u_{m-1}) \in \{0, 1\}^{m-1}$. Then

$$R = (P_1 + u_1 T_2) + (P_2 + u_2 T_2) + \cdots + (P_{m-1} + u_{m-1} T_2) + \left(P_m + \left(\sum_{i=1}^{m-1} u_i\right) T_2\right). \tag{3.5}$$

This gives 2^{m-1} different decompositions. Since $t(P_i) = t(P_i + T_2)$, it follows that if $t_1, t_2, \cdots, t_m$ is a solution to the summation polynomial corresponding to the original decomposition then the same solution is true for all 2^{m-1} decompositions (3.5). We wish to have non-unique solutions so that we reduce size of the solution set by applying a change of variables thereby reducing the degree in the ideal of the corresponding polynomial system. But this has been taken care of by the design of the function t. So the FGLM complexity is already reduced.

To speed up the linear algebra, the size of the factor base can be reduced. Each solution $(t_1, \ldots, t_m)$ corresponds to many relations. Let us fix, for each $t(P)$, one of the four points $\{P, -P, P + T_2, -P + T_2\}$, and put only that point into our factor base. As a result the size of $\mathcal{F}$ is exactly the same as the number of $t(P) \in V$ that correspond to elliptic curve points, which is roughly $\frac{1}{4}\#V$.

So for a point R, given a solution $f_{m+1}(t_1, \ldots, t_m, t(R)) = 0$ there is a unique value $z_0 \in \{0, 1\}$, unique points $P_1, \ldots, P_m \in \mathcal{F}$, and unique choices of sign $z_1, \ldots, z_m \in \{-1, 1\}$ such that

$$R + z_0 T_2 = \sum_{i=1}^{m} z_i P_i.$$

It follows that the size of the matrix we build after collecting enough relations is reduced by a factor of $1/4$ (with one extra column added to store the coefficient of T_2). This means we need to find fewer relations and the complexity of the linear algebra is reduced by a factor of $(1/4)^2$.

3.2.3 The action of points of order 4

We now consider the binary Edwards curve E_{d_1,d_2} in the case $d_1 = d_2$. We observe that $T_4 = (1, 0) \in E_{d_1,d_2}$ and one can verify that $T_4 + T_4 = (1, 1) = T_2$ and so T_4 has order four. The group generated by T_4 is therefore $\{P_0, T_4, T_2, -T_4 = (0, 1)\}$.

For a point $P = (x, y) \in E_{d_1,d_2}$, we have $P+T_4 = (y, x+1)$. So $t(P+T_4) = t(P)+1$. We construct our factor base $\mathcal{F}$ such that $(x, y) \in \mathcal{F}$ implies $(y, x + 1) \in \mathcal{F}$. For example, we can choose a vector subspace $V \subseteq \mathbb{F}_{2^n}$ such that $v \in V$ if and only if $v+1 \in V$ and set $\mathcal{F} = \{P \in E_{d_1,d_2}(\mathbb{F}_{2^n}) \mid t(P) \in V\}$.

Let $R = P_1 + P_2 + \cdots + P_m$, where $P_i \in \mathcal{F}$. Let $(u_1, \ldots, u_{m-1}) \in \{0, 1, 2, 3\}^{m-1}$ then we get

$$R = (P_1 + [u_1]T_4) + (P_2 + [u_2]T_4) + \cdots + (P_{m-1} + [u_{m-1}]T_4) + (P_m + [-\sum_{i=1}^{m-1} u_i]T_4), \quad (3.6)$$

decompositions of R as a sum of m factor base elements for free. This gives 4^{m-1} distinct decompositions. A factor of 2^{m-1} of these decompositions are due to the action of T_2, taking $u_i \in \{0, 2\}$. The remaining 2^{m-1} factors of these decompositions are due to the action of T_4 and $-T_4$. So if $t_1, t_2, \cdots, t_m$ is a solution to the original decomposition, then we get 2^{m-1} distinct solutions of the form $(t_1 + r_1, t_2 + r_2, \cdots, t_m + \sum_{i=1}^{m-1} r_i)$ for $(r_1, \ldots, r_{m-1}) \in \{0, 1\}^{m-1}$. This corresponds to an even number of times, addition of 1 to the original solution.

We observe that the dihedral coxeter group $D_m = (\mathbb{Z}/2\mathbb{Z})^{m-1} \rtimes S_m$ acts on the summation polynomial, where $(\mathbb{Z}/2\mathbb{Z})^{m-1}$ adds 1 an even number of times to the solutions of the summation polynomial and S_m permutes them. This group leaves invariant the summation polynomials. So we have

$$f_{m+1}(t_1, t_2, \ldots, t_m, t(R)) \in \mathbb{F}_{2^n}[t_1, t_2, \ldots, t_m]^{D_m}.$$

To express the summation polynomial in terms of the generators of the invariant ring $\mathbb{F}_{2^n}[t_1, t_2, \ldots, t_m]^{D_m}$, it suffices to note that the invariants under the map $t_i \mapsto t_i + 1$ in characteristic 2 are $t_i(t_i + 1) = t_i^2 + t_i$ (this is mentioned in Section 4.3 of [FHJ+14]). So we construct such invariant variables under this map using the elementary symmetric polynomials in the variables $t_i^2 + t_i$,

$$\begin{aligned} s_2 &= (t_1^2 + t_1)(t_2^2 + t_2) + \cdots + (t_{m-1}^2 + t_{m-1})(t_m^2 + t_m), \\ &\vdots \\ s_m &= (t_1^2 + t_1)(t_2^2 + t_2) \cdots (t_m^2 + t_m). \end{aligned} \quad (3.7)$$

One might also expect to use

$$s_1 = t_1 + t_1^2 + \cdots + t_m + t_m^2 = e_1 + e_1^2,$$

where $e_1 = t_1 + \cdots + t_m$. But since the addition by T_4 cancels out in equation (3.6), we actually have that e_1 remains invariant. Thus we use the invariant variables $e_1, s_2, \ldots, s_m$ as the generators of the invariant ring $\mathbb{F}_{2^n}[t_1, t_2, \ldots, t_m]^{D_m}$.

Rewriting the summation polynomial in terms of the new variables reduces the number of solutions by the order of the group D_m. This in turn reduces the degree of the ideal of the corresponding polynomial system by the same amount. As a result, the complexity of the FGLM algorithm (see Theorem 2.1.19) is reduced by a factor of $(m!2^{m-1})^3$.

It is clear that we further halve the size of the factor base by choosing a unique representative of the orbit under the action. Note that a solution to the summation polynomial equation exists if there exists a unique value $z_0 \in \{0, 1, 2, 3\}$, unique points $P_1, \ldots, P_m \in \mathcal{F}$, and unique choices of sign $z_1, \ldots, z_m \in \{-1, 1\}$ such that

$$R + z_0 T_4 = \sum_{i=1}^{m} z_i P_i.$$

Overall due to the action of the symmetric group S_m of order $m!$ and the point T_4 of order 4, the factor base is reduced in total by a factor of $1/8$. As a result the complexity of the linear algebra is reduced by a factor of $(1/8)^2$.

3.3 Index calculus algorithm

We now present the full index calculus algorithm (see Algorithm 1) combined with the new variables introduced in Section 3.2.3. We work in $E(\mathbb{F}_{2^n}) = E_{d_1,d_2}(\mathbb{F}_{2^n})$ where n is prime and E_{d_1,d_2} is a binary Edwards curve with parameters $d_2 = d_1$. Since n is prime, we are trying to solve the polynomial system of equations given by equation (3.3).

For the linear algebra stage of the index calculus algorithm, we will require $\#\mathcal{F} + 1 \approx \#V = 2^l$ relations of the form

$$u_j P + w_j Q = z_{j,0} T_4 + \sum_{P_i \in \mathcal{F}} z_{j,i} P_i,$$

where $M = (z_{j,i})$ is a sparse matrix with at most m non-zero entries per row, $u_j P + w_j Q = R$ is a random element for known random integer values of u_j and w_j, and $z_{j,0} \in \{0, 1, 2, 3\}$.

If we can find a solution $(t_1, t_2, \ldots, t_m) \in V^m$ satisfying $f_{m+1}(t_1, t_2, \ldots, t_m, t(R)) = 0$ then we need to determine the corresponding points, if they exist, $(x_i, y_i) \in E(\mathbb{F}_{2^n})$ such that $t_i = x_i + y_i$ and $(x_1, y_1) + \cdots + (x_m, y_m) = R$. Finding (x_i, y_i) given t_i is just taking roots of a univariate quartic polynomial. Once we have m points in $E(\mathbb{F}_{2^n})$, we may need to check up to 2^{m-1} choices of sign and also determine an additive term $z_{j,0}T_4$ to be able to record the relation as a vector. The cost of computing the points (x_i, y_i) is almost negligible, but checking the signs may incur some cost for large m.

If a random point R can be written as a sum of m factor base elements, then there exists a solution $(t_1, \ldots, t_m) \in V^m$ to the polynomial system that can be lifted to points in $E(\mathbb{F}_{2^n})$. When no relation exists there are two possible scenarios: Either there is no solution $(t_1, \ldots, t_m) \in V^m$ to the polynomial system, or there are solutions but they do not lift to points in $E(\mathbb{F}_{2^n})$.

Let r be the order of P (assumed to be odd). If S is any vector in the kernel of the matrix M, that is $SM \equiv 0 \pmod{r}$), then write $u = S(u_1, \ldots, u_{\ell+1})^T$ and $w = S(w_1, \ldots, w_{\ell+1})^T$. We have $uP + wQ = 0$ (the T_4 term must disappear if r is odd) and so $u + wa \equiv 0 \pmod{r}$. A solution to the ECDLP is then a.

We now detail how to use the invariant variables (3.7) to speed up the resolution of solving the system of polynomial equations (3.3). From our discussion we have $t(P_j) = \sum_{i=0}^{\ell-1} c_{j,i}\theta^i$ (3.2) for $1 \leq j \leq m$. We have observed that the polynomial system has $n + \ell m$ equations and ℓm binary variables $c_{j,i}$. The system of polynomial equations (3.3) is obtained by first substituting $t(P_j)$ and then applying Weil descent to the summation polynomials. This time instead of substituting t_j, we make use of the invariant variables. Let us drop the notation $t(P_j)$ and write instead t_j.

As noted by Huang et al. [HPST13], write the invariant variables e_1 and s_j in terms of binary variables with respect to the basis for $\mathbb{F}_{2^n}$. We find the smallest vector subspace in terms of the dimension of the vector subspace V of $\mathbb{F}_{2^n}$ that contains each invariant variable. For example $e_1 = t_1+t_2+\cdots+t_m$. Since each $t_j \in V$ and the operation of addition is closed in the vector subspace V. It follows that $e_1 \in V$ and it can be written as

$$e_1 = d_{1,0} + d_{1,1}\theta + d_{1,2}\theta^2 + \cdots + d_{1,\ell-1}\theta^{\ell-1}, \tag{3.8}$$

where $d_{i,j} \in \mathbb{F}_2$. Similarly the invariant variables $s_2, \ldots, s_m$ can be written as,

$$\begin{aligned}
s_2 &= d_{2,0} + d_{2,1}\theta + d_{2,2}\theta^2 + \cdots + d_{2,4(\ell-1)}\theta^{4(\ell-1)},\\
&\vdots\\
s_j &= d_{j,0} + d_{j,1}\theta + d_{j,2}\theta^2 + \cdots + d_{j,2j(\ell-1)}\theta^{2j(\ell-1)}\\
&\quad \text{where } 1 \le j \le k = \min(\lfloor n/(2(\ell-1))\rfloor, m),\\
s_{j+1} &= d_{j+1,0} + d_{j+1,1}\theta + d_{j+1,2}\theta^2 + \cdots + d_{j+1,(n-1)}\theta^{n-1},\\
&\vdots\\
s_m &= d_{m,0} + d_{m,1}\theta + d_{m,2}\theta^2 + \cdots + d_{m,n-1}\theta^{n-1}.
\end{aligned} \tag{3.9}$$

Suppose $k = n/(2(\ell-1)) \approx m/2$ and it takes the value $\tilde{m} = \lceil m/2 \rceil$, where $\lceil m/2 \rceil$ denotes the smallest integer greater or equal to $m/2$. Then the number of binary variables $d_{i,j}$ is

$$N = \ell + (4(\ell-1)+1) + (6(\ell-1)+1) + \cdots + (2\tilde{m}(\ell-1)+1) + \tilde{m}n \approx (m^2\ell + mn)/2.$$

Let $\tilde{f}_m$ be f_{m+1} evaluated at $t(R)$. Then substituting $e_1, s_2, \cdots, s_m$ in $\tilde{f}_m$ we obtain

$$\begin{aligned}
0 &= \tilde{f}_m(t_1, t_2, \cdots, t_m),\\
&= \tilde{f}_m(e_1, s_2, \cdots, s_m),\\
&= \tilde{f}_m\Big(\sum_{j=0}^{\ell-1} d_{1,j}\theta^j, \cdots, \sum_{j=0}^{n-1} d_{m,j}\theta^j\Big),\\
&= \sum_{i=0}^{n-1} \phi_i(d_{1,0}, d_{1,1}\cdots, d_{m,n-2}, d_{m,n-1})\theta^i,\\
\iff & \phi_1(d_{1,0}, \cdots, d_{m,n-1}) = \cdots = \phi_{n-1}(d_{1,0}, \cdots, d_{m,n-1}) = 0,
\end{aligned} \tag{3.10}$$

where $\phi_i \in \mathbb{F}_2[d_{1,0}, \cdots, d_{m,n-1}]$. This forms a system of n equations in the N binary variables $d_{j,i}$. We add the field equations $d_{j,i}^2 - d_{j,i}$. Denote the system of equations by sys_1.

One could attempt to solve sys_1 using the F4 or F5 Gröbner basis algorithms. But solving sys_1 is harder than solving the original polynomial system of equations (3.3) as the number of binary variables N is too large compared with the number of equations. Therefore, we add a large number of new equations to sys_1 relating the $c_{j,\tilde{i}}$ to the $d_{j,i}$ via the t_j, using equations (3.7) and (3.2). But this also introduces
$\ell m < n$ variables $c_{j,\tilde{i}}$. So for the invariant variables $e_1, s_2, \cdots, s_m$, we get the relations

$$\begin{aligned}
d_{1,0} + d_{1,1}\theta + \cdots + d_{1,\ell-1}\theta^{\ell-1} &= \sum_{j=1}^{m}\sum_{\tilde{i}=0}^{\ell-1} c_{j,\tilde{i}}\theta^{\tilde{i}},\\
&\vdots\\
d_{1,0} + d_{1,1}\theta + \cdots + d_{1,n-1}\theta^{n-1} &= \prod_{j=1}^{m}\sum_{\tilde{i}=0}^{\ell-1} c_{j,\tilde{i}}\theta^{\tilde{i}},
\end{aligned}$$

respectively. Applying Weil descent to this gives N additional equations in the $N+\ell m$ binary variables. After adding the field equations $c_{j,\tilde{i}}^2 - c_{j,\tilde{i}}$, we denote the system of equations by sys_2.

The combined system of equations $sys = sys_1 \cup sys_2$ has $n + N$ equations, $N + \ell m$ field equations and $N + \ell m$ binary variables. So sys is an overdetermined system. Finally we solve sys using the F4 or F5 Gröbner basis algorithms. If a solution to the systems of equations exists, for each candidate solution $c_{j,\tilde{i}}$, one computes the corresponding solution $(t_1, t_2, \ldots, t_m)$. For each t_j the corresponding point P_j is computed and the points $P_1, P_2, \cdots, P_m$ are checked if they form a relation. If they do, we record the relation and continue the operation until we get enough relations for the linear algebra stage.

Algorithm 1 Index Calculus Algorithm

1: Set $N_r \leftarrow 0$
2: **while** $N_r \leq \#\mathcal{F}$ **do**
3: Compute $R \leftarrow uP + wQ$ for random integer values u and w.
4: Compute the summation polynomial with respect to invariant variables, $f_{m+1}(e_1, s_2, \ldots, s_m, t(R))$.
5: Evaluate the summation polynomial f_{m+1} at $t(R)$ to get $\tilde{f}_m$.
6: Use Weil descent to write $\tilde{f}_m(e_1, s_2, \ldots, s_m)$ as n polynomial equations in the binary variables $d_{j,i}$.
7: Add the field equations $d_{j,i}^2 - d_{j,i}$ to get system of equations sys_1.
8: Build new polynomial equations relating the variables $d_{j,i}$ and $c_{j,\tilde{i}}$.
9: Add field equations $c_{j,\tilde{i}}^2 - c_{j,\tilde{i}}$ to get system of equations sys_2.
10: Solve system of equations $sys = sys_1 \cup sys_2$ to get $(c_{j,\tilde{i}}, d_{j,i})$.
11: Compute corresponding solution(s) $(t_1, \ldots, t_m)$.
12: For each t_j compute, if it exists, a corresponding point $P_j = (x_j, y_j) \in \mathcal{F}$
13: **if** $z_1P_1 + z_2P_2 + \cdots + z_mP_m + z_0T_4 = R$ for suitable $z_0 \in \{0, 1, 2, 3\}, z_i \in \{1, -1\}$ **then**
14: $N_r \leftarrow N_r + 1$
15: Record z_i, u, w in a matrix M for the linear algebra
16: Use linear algebra to find non-trivial kernel element to solve the ECDLP.

3.4 Breaking symmetry in the factor base

If $\#V^m \approx 2^n$, the probability of decomposing a random element R as a sum of m factor base elements is approximately $\frac{1}{m!}$. If m is large then the probability of finding a relation is low. Although in practice for small m the running time of the index calculus algorithm is extremely slow, it is worth investigating how to increase the probability of finding a relation to approximately equal to 1. The probability of finding a relation is approximately 1 means, the cost of finding a relation is reduced roughly by $m!$.

We now explain how to break symmetry in the factor base while using the invariant variables (3.7). This is our greedy approach to lower the probability of finding a relation as well as enhancing the resolution of solving the system of equations sys (see line 10 of Algorithm 1).

Suppose $\mathbb{F}_{2^n}$ is represented using a polynomial basis and take V to be the subspace with basis $\{1, \theta, \ldots, \theta^{\ell-1}\}$. We choose m elements $v_i \in \mathbb{F}_{2^n}$ (which can be interpreted as vectors in the n-

dimensional $\mathbb{F}_2$-vector space corresponding to $\mathbb{F}_{2^n}$) as follows:

$$\begin{aligned}
v_1 &= 0, \\
v_2 &= \theta^\ell = (0,0,\ldots,0,1,0,\ldots,0) \quad \text{where the 1 is in position } \ell \text{ Similarly ,} \\
v_3 &= \theta^{\ell+1}, \\
v_4 &= \theta^{\ell+1}+\theta^\ell, \\
v_5 &= \theta^{\ell+2}, \\
&\vdots
\end{aligned}$$

In other words, v_i is represented as a vector of the form $(0,\ldots,0,w_0,w_1,\ldots)$ where $\ldots w_1 w_0$ is the binary expansion of $i-1$. Note that the subsets $V+v_i$ in $\mathbb{F}_{2^n}$ are pair-wise disjoint.

Accordingly, we define the factor bases to be

$$\mathcal{F}_i = \{P \in E(\mathbb{F}_{2^n}) \mid t(P) \in V + v_i\} \quad \text{for } 1 \leq i \leq m,$$

where $t(x,y) = x+y$. A similar factor base design is mentioned in Section 7 of [Nag13] to solve DLP over hyper elliptic curves.

The decomposition over the factor base of a point R will be a sum of the form $R = P_1+P_2+\cdots+P_m$ where $P_i \in \mathcal{F}_i$ for $1 \leq i \leq m$. Since we heuristically assume that $\#\mathcal{F}_i \approx 2^\ell$, we expect the number of points in $\{P_1+\cdots+P_m \mid P_i \in \mathcal{F}_i\}$ to be roughly $2^{\ell m}$. Note that there is no $1/m!$ term here. The entire purpose of this definition is to break the symmetry in the factor base to increase the probability of relations. So the probability that a uniformly chosen point $R \in E(\mathbb{F}_{2^n})$ can be decomposed in this way is heuristically $2^{\ell m}/2^n = 1/2^{n-\ell m}$, which is approximately 1 for $n \approx \ell m$.

As the points P_i are chosen from distinct factor bases $\mathcal{F}_i$ in the decomposition $R = P_1+\cdots+P_m$, one does not have the action by the symmetric group S_m. But the summation polynomials have the action by S_m. So they can be written in terms of the invariant variables $e_1, s_2, \cdots, s_m$. The trick is to distinguish the computation of the invariant variables during the construction of the system of equations sys via Weil descent.

Take for example $m=4$, we are decomposing a random element R as a sum of 4 factor base elements ($R = P_1+P_2+P_3+P_4$). So one needs to solve the summation polynomial $f_5(t_1,t_2,t_3,t_4,t(R))$. Without breaking the symmetry in the factor base, we have $t_j \in V$ which can be written as $t_j = \sum_{\tilde{i}=0}^{\ell-1} c_{j,\tilde{i}}\theta^{\tilde{i}}$. But with the symmetry breaking of the factor base we have

$$\begin{aligned}
t_1 &= c_{1,0}+c_{1,1}\theta+c_{1,2}\theta^2+\cdots+c_{1,\ell-1}\theta^{\ell-1}, \\
t_2 &= c_{2,0}+c_{2,1}\theta+c_{2,2}\theta^2+\cdots+c_{2,\ell-1}\theta^{\ell-1}+\theta^\ell, \\
t_3 &= c_{3,0}+c_{3,1}\theta+c_{3,2}\theta^2+\cdots+c_{3,\ell-1}\theta^{\ell-1}+\theta^{\ell+1}, \\
t_4 &= c_{4,0}+c_{4,1}\theta+c_{4,2}\theta^2+\cdots+c_{4,\ell-1}\theta^{\ell-1}+\theta^\ell+\theta^{\ell+1}.
\end{aligned}$$

It follows that the invariant variable

$$e_1 = t_1+t_2+t_3+t_4 = d_{1,0}+d_{1,1}\theta+\cdots+d_{1,\ell-1}\theta^{\ell-1}$$

can be represented exactly as in (3.8). But the other invariant variables are less simple. For example,

$$\begin{aligned}
s_2 &= (t_1^2+t_1)(t_2^2+t_2)+\cdots+(t_3^2+t_3)(t_4^2+t_4) \\
&= d_{2,0}+d_{2,1}\theta+d_{2,2}\theta^2+\cdots+d_{2,4(\ell-1)}\theta^{4(\ell-1)}+\cdots+\theta^{4\ell+4}.
\end{aligned}$$

Without breaking the symmetry in the factor base s_2 has the highest term $d_{2,4(\ell-1)}\theta^{4\ell-4}$ (3.9) but now with breaking the symmetry in the factor base it has highest terms $d_{2,4(\ell+1)}\theta^{4\ell+4}$. The number of variables has increased by 8. But most of them are either 0 or 1 and they can be fixed during the Weil descent. For example the coefficient of $\theta^{4\ell+4}$ is 1. So $d_{2,4(\ell+1)} = 1$. In general we require more variables than with the symmetry in the factor base case. So breaking the symmetry in the factor base increases the probability of a relation but produces a harder system of equations to solve.

An additional consequence of this idea is that the factor base is now roughly m times larger than in the symmetric case. So the number of relations required is increased by a factor m. So the speed up actually is approximately $m!/m = (m-1)!$ and the cost of the linear algebra is also increased by a factor m^2.

To determine the usefulness of breaking the symmetry in the factor base, it is necessary to perform some experiments (see Section 3.6). As our experiment shows, breaking the symmetry in the factor base gives an advantage as n gets larger and the size of the factor base is small.

3.5 Gröbner basis versus SAT solvers comparison

Shantz and Teske [ST13] discuss a standard idea [YC04, YCC04, BFP09] called the "hybrid method", which is to partially evaluate the system at some random points before applying the F4 or F5 Gröbner basis algorithms. It is argued that it is better to just use the "delta method", $\Delta = n - m\ell > 0$. The main observation is that using smaller ℓ speeds up the Gröbner basis computation at the cost of decreasing the probability of getting a relation.

We investigated other approaches to solve polynomial systems of equations over a binary field. In particular, we experimented with SAT solvers. We used Minisat 2.1 [SE08] (see also [ES, SE05]), coupled with the Magma system for converting the polynomial system into conjunctive normal form (CNF).

SAT solvers take an input in Conjunctive Normal Form (CNF): a conjunction of clauses where a clause is a disjunction of literals, and a literal is a variable or its negation. The Magma interface with Minisat performs the conversion from polynomial equations to CNF. The number of variables, the number of clauses, and the total length of all the clauses determines the size of the CNF expression. Although the running time of SAT solvers in the worst case is exponential in the number of variables in the problem, practical running times may be shorter as we will see in our experimental Section 3.6.

3.6 Experimental results

We conducted several experiments using elliptic curves E over $\mathbb{F}_{2^n}$. We always use the $(m+1)^{th}$ summation polynomial to find relations as a sum of m points in the factor base. The factor base is defined using a vector space of dimension ℓ. In our experiments we examine the effect of the invariant variables $e_1, s_2, \ldots, s_m$ on the computation of intermediate results and the degree of regularity d_{reg}. Recall that d_{reg} is the main complexity indicator of the F4 or F5 Gröbner basis algorithms. The time and memory complexities are roughly estimated to be $N^{3d_{\text{reg}}}$ and $N^{2d_{\text{reg}}}$ respectively where N is the number of variables.

Experiment 1: For the summation polynomials we use the invariant variables $e_1, e_2, \ldots, e_m$, which are the generators of the invariant ring under the group S_m (the action of T_2 is exploited in this experiment). The factor base is defined with respect to a fixed vector space of dimension ℓ.

Experiment 2: For the summation polynomials we use the invariant variables $e_1, s_2, \ldots, s_m$, which are the generators of the invariant ring under the group $D_m = (\mathbb{Z}/2\mathbb{Z})^{m-1} \rtimes S_m$ (note the action of T_4 is exploited in this experiment). The factor base is defined with respect to a fixed vector space V of dimension ℓ such that $v \in V$ if and only if $v + 1 \in V$.

Experiment 3: For the summation polynomials we use the invariant variables $e_1, s_2, \ldots, s_m$, which are generators of the invariant ring under the group $(\mathbb{Z}/2\mathbb{Z})^{m-1} \rtimes S_m$. The symmetry in the factor base is broken. We define the factor base by taking affine spaces (translations of a vector space of dimension ℓ).

We denote the time taken for the set-up operations (lines 4 to 9 of Algorithm 1) by T_{Inter}, while T_{GB} denotes the time taken to do line 10 of the algorithm. Other notation includes Mem (the average memory used in megabytes by the Minisat SAT solver or Gröbner basis), d_{reg} (the degree of regularity), Var (the total number of variables in the system) and P_{equ} (the total number of equations).

In Table 3.1 we also give a success probability P_{succ} the percentage of times our SAT program terminated with solution within 200 seconds, T_{SAT} the average running times in seconds to compute step 10 using a SAT solver, and #Clauses and #Literals are the average number of clauses and total number of literals (i.e., total length) of the CNF input to the SAT solver.

All experiments are carried out using a computational server (3.0GHz CPU x8, 28G RAM). In all our experiments, timings are averages of 100 trials except for values of $T_{\text{GB}} + T_{\text{Inter}} > 200$ seconds (our patience threshold), in this case they are single instances.

Table 3.1 compares Minisat with Gröbner basis methods (Experiment 2) for $m = 4$. The experiment shows that SAT solvers can be faster and, more importantly, handle larger range of values for ℓ. As is shown in Table 3.1, we can work with ℓ up to 7. But the F4 or F5 Gröbner basis algorithms are limited to $\ell \in \{3, 4\}$ for fixed value $m = 4$ in our experiments.

However, on the negative side, the running time of SAT solvers varies a lot depending on many factors such as the curve parameter d_1, and the hamming weight of possible solutions. Further, our experimental investigation shows that they seem to be faster when there is a solution of low hamming weight. We also observe from our experiment that SAT solvers are slightly slow when no solution exists. This behavior is very different to the case of Gröbner basis methods, which perform rather reliably and are slightly better when our system of equations have no solution. For both methods, Table 3.1 shows only the case when our system of equations have a solution.

SAT solvers with an "early abort" strategy gives an advantage. That is one can generate a lot of instances and run SAT solvers in parallel and then kill all instances that are still running after some time threshold has been passed (a similar idea is mentioned in Section 7.1 of [MCP07]). This could allow the index calculus algorithm to be run for a larger set of parameters. The probability of finding a relation is now decreased. The probability that a relation exists must be multiplied by the probability that the SAT solver terminates in less than the time threshold, in the case when a solution exists. It is this latter probability that we estimate in the P_{succ} column of Table 3.1.

Table 3.2 compares Experiment 1 and Experiment 2 in the case $m = 3$. Gröbner basis methods are used in both cases. Timings are averages from 100 trials except for values of $T_{GB} + T_{Inter} > 200$ seconds, in this case they are single instances.

Experiments in [HPST13] are limited to the case $m = 3$ and $\ell \in \{3, 4, 5, 6\}$ for prime degree extensions

$$n \in \{17, 19, 23, 29, 31, 37, 41, 43, 47, 53\}.$$

This is due to high running times and large memory requirements, even for small parameter sizes. As shown in Table 3.2, we repeated these experiments. Exploiting greater symmetry (in this case Experiment 2) is seen to reduce the computational costs. Indeed, we can go up to $\ell = 8$ with reasonable running time for some n, which is further than [HPST13]. The degree of regularity stays ≤ 4 in both cases.

Table 3.4 considers $m = 4$, which was not done in [HPST13]. For the sake of comparison, we gather some data for Experiment 1 and Experiment 2. Again, exploiting greater symmetry (Experiment 2) gives a significant decrease in the running times, and the degree of regularity d_{reg} is slightly decreased. The expected degree of regularity for $m = 4$, stated in [PQ12], is $m^2 + 1 = 17$. The table shows that our choice of coordinates makes the case $m = 4$ much more feasible.

Table 3.1: Comparison of solving polynomial systems, when there exists a solution to the system, in Experiment 2 using SAT solver (Minisat) versus Gröbner basis methods for $m = 4$. $\#\mathrm{Var}$ and $\#P_{\mathrm{equ}}$ are the number of variables and the number of polynomial equations respectively. Mem is average memory used in megabytes by the SAT solver or Gröbner basis algorithm. $\#\mathrm{Clauses}$, $\#\mathrm{Literals}$, and P_{succ} represent the average number of clauses, total number of literals, and the percentage of times Minisat halts with solutions within 200 seconds respectively.

Experiment 2 with SAT solver Minisat									
n	ℓ	$\#\mathrm{Var}$	$\#P_{\mathrm{equ}}$	$\#\mathrm{Clauses}$	$\#\mathrm{Literals}$	T_{Inter}	T_{SAT}	Mem	P_{succ}
17	3	54	59	46678	181077	0.35	7.90	5.98	94%
	4	67	68	125793	485214	0.91	27.78	9.38	90%
19	3	54	61	55262	215371	0.37	3.95	6.07	93%
	4	71	74	140894	543422	1.29	18.38	18.05	86%
23	3	54	65	61572	240611	0.39	1.53	7.60	87%
	4	75	82	194929	760555	2.15	5.59	14.48	83%
	5	88	91	394759	1538560	4.57	55.69	20.28	64%
29	4	77	90	221828	868619	3.01	7.23	19.05	87%
	5	96	105	572371	2242363	9.95	39.41	32.87	67%
	6	109	114	855653	3345987	21.23	15.87	43.07	23%
	7	118	119	1063496	4148642	36.97	26.34	133.13	14%
31	4	77	92	284748	1120243	3.14	17.12	20.52	62%
	5	98	109	597946	2345641	11.80	33.48	45.71	57%
	6	113	120	892727	3489075	26.23	16.45	118.95	12%
	7	122	125	1307319	5117181	44.77	21.98	148.95	8%
37	4	77	98	329906	1300801	3.41	26.12	29.97	59%
	5	100	117	755621	2977220	13.58	48.19	50.97	40%
	6	119	132	1269801	4986682	41.81	42.85	108.41	11%
	7	134	143	1871867	7350251	94.28	40.15	169.54	6%
41	4	77	102	317272	1250206	3.08	19.28	27.59	68%
	5	100	121	797898	3146261	15.71	27.14	49.34	65%
	6	123	140	1353046	5326370	65.25	31.69	89.71	13%
43	4	77	104	374011	1477192	2.97	17.77	28.52	68%
	5	100	123	825834	3258080	13.85	29.60	54.83	52%
47	4	77	108	350077	1381458	3.18	11.40	29.93	59%
	5	100	127	836711	3301478	14.25	27.56	61.55	43%
53	4	77	114	439265	1738168	11.02	27.88	32.35	75%
	5	100	133	948366	3748119	14.68	34.22	64.09	62%
	6	123	152	1821557	7200341	49.59	41.55	123.38	11%
	7	146	171	2930296	11570343	192.20	67.27	181.20	4%

Experiment 2 with Gröbner basis: F_4						
n	ℓ	$\#\mathrm{Var}$	$\#P_{\mathrm{equ}}$	T_{Inter}	T_{GB}	Mem
17	3	54	59	0.29	0.29	67.24
	4	67	68	0.92	51.79	335.94
19	3	54	61	0.33	0.39	67.24
	4	71	74	1.53	33.96	400.17
23	3	54	65	0.26	0.31	67.24
	4	75	82	2.52	27.97	403.11
29	3	54	71	0.44	0.50	67.24
	4	77	90	3.19	35.04	503.87
31	3	54	73	0.44	0.58	67.24
	4	77	92	3.24	9.03	302.35
37	3	54	79	0.36	0.43	67.24
	4	77	98	3.34	9.07	335.94
41	3	54	83	0.40	0.54	67.24
	4	77	102	3.39	17.19	382.33
43	3	54	85	0.43	0.53	67.24
	4	77	104	3.44	9.09	383.65
47	3	54	89	0.50	0.65	67.24
	4	77	108	3.47	9.59	431.35
53	3	54	95	0.33	0.40	67.24
	4	77	114	11.43	11.64	453.77

Table 3.2: Comparison of solving our systems of equations, having a solution, using Gröbner basis methods in Experiment 1 and Experiment 2 for $m = 3$. Notation is as above. '*' indicates that the time to complete the experiment exceeded our patience threshold

Experiment 1						
n	ℓ	d_{reg}	#Var	$\#P_{\mathrm{equ}}$	T_{Inter}	T_{GB}
17	5	4	42	44	0.08	13.86
19	5	4	42	46	0.08	18.18
	6	4	51	52	0.18	788.91
23	5	4	42	50	0.10	35.35
	6	4	51	56	0.21	461.11
	7	*	*	*	*	*
29	5	4	42	56	0.11	31.64
	6	4	51	62	0.25	229.51
	7	4	60	68	0.60	5196.18
	8	*	*	*	*	*
31	5	4	42	58	0.12	5.10
	6	5	51	64	0.27	167.29
	7	5	60	70	0.48	3259.80
	8	*	*	*	*	*
37	5	4	42	64	0.18	0.36
	6	4	51	70	0.34	155.84
	7	4	60	76	0.75	1164.25
	8	*	*	*	*	*
41	5	4	42	68	0.16	0.24
	6	4	51	74	0.36	251.37
	7	4	60	80	0.77	1401.18
	8	*	*	*	*	*
43	5	4	42	70	0.19	0.13
	6	4	51	76	0.38	176.67
	7	3	60	82	0.78	1311.23
	8	*	*	*	*	*
47	5	4	42	74	0.19	0.14
	6	4	51	80	0.54	78.43
	7	*	*	*	*	*
	8	*	*	*	*	*
53	5	4	51	80	0.22	0.19
	6	5	51	86	0.45	1.11
	7	4	60	92	1.20	880.59
	8	*	*	*	*	*

Experiment 2						
n	ℓ	d_{reg}	#Var	$\#\mathrm{P}_{\mathrm{eq}}$	T_{Inter}	T_{GB}
17	5	4	54	56	0.02	0.41
19	5	3	56	60	0.02	0.48
	6	4	62	63	0.03	5.58
23	5	4	60	68	0.02	0.58
	6	4	68	73	0.04	2.25
	7	*	*	*	*	*
29	5	4	62	76	0.03	0.12
	6	4	74	85	0.04	2.46
	7	4	82	90	0.07	3511.14
	8	*	*	*	*	*
31	5	4	62	78	0.03	0.36
	6	4	76	89	0.05	2.94
	7	4	84	94	0.07	2976.97
	8	*	*	*	*	*
37	5	4	62	84	0.04	0.04
	6	4	76	95	0.06	4.23
	7	4	90	106	0.09	27.87
	8	*	*	*	*	*
41	5	4	62	88	0.03	0.04
	6	4	76	99	0.06	0.49
	7	4	90	110	0.09	11.45
	8	*	*	*	*	*
43	5	3	62	90	0.04	0.05
	6	4	76	101	0.06	5.35
	7	4	90	112	0.10	15.360
	8	*	*	*	*	*
47	5	4	62	94	0.04	0.06
	6	4	76	105	0.06	1.28
	7	4	90	116	0.13	8.04
	8	4	104	127	0.16	152.90
53	5	3	62	100	0.04	0.02
	6	4	76	111	0.06	0.19
	7	4	90	122	0.14	68.23
	8	4	104	133	0.19	51.62

Our idea of breaking symmetry in the factor base (Experiment 3) is investigated in Table 3.3 for the case $m = 3$. Some of the numbers in the second tabular column already appeared in Table 3.2. Recall that the relation probability is increased by a factor $3! = 6$ in this case, so one should multiply the timings in the right hand column by $(m-1)! = 2$ to compare overall algorithm speeds. The experiments are not fully conclusive (and there are a few "outlier" values that should be ignored), but it shows that breaking the symmetry in the factor base gives a speedup in many cases when n is large.

For larger values of n, the degree of regularity d_{reg} is often 3 for breaking the symmetry in the factor base case while it is 4 for most values in Experiment 2. So the performance we observe with breaking the symmetry in the factor base is explained by the fact that the degree of regularity stayed at 3 as n grows.

Conclusion

We conclude that cryptosystems making use of the DLP for binary curves as proof of their security are safe. As the above experiments show, our index calculus algorithm is limited to very small range of parameters. So the Pollard rho/lambda and its variants remain the best algorithms to solve the DLP for binary curves.

Table 3.3: Comparison of solving our systems of equations using Gröbner basis methods having a solution in Experiment 3 and Experiment 2 when $m = 3$. Notation is as in Table 3.1. For a fair comparison, the timings in the right hand column should be doubled.

Experiment 3						
n	ℓ	d_{reg}	#Var	$\#P_{\text{equ}}$	T_{Inter}	T_{GB}
37	5	3	68	90	0.04	0.25
	6	4	80	99	0.07	5.67
	7	*	*	*	*	*
41	5	4	68	94	0.05	0.39
	6	3	80	103	0.07	4.55
	7	4	93	113	0.11	1905.21
43	5	4	68	96	0.05	0.21
	6	4	80	105	0.08	4.83
	7	3	94	116	0.12	100.75
47	5	4	68	100	0.05	0.17
	6	3	80	109	0.08	3.88
	7	3	94	120	0.11	57.61
53	5	3	68	106	0.06	0.08
	6	4	80	115	0.09	12.75
	7	3	94	126	0.14	11.38
59	5	4	68	112	0.06	0.05
	6	4	80	121	0.10	0.59
	7	4	94	132	0.16	13.60
61	5	4	68	114	0.06	0.04
	6	4	80	123	0.11	0.46
	7	4	94	134	0.16	8.61
67	5	3	68	120	0.07	0.02
	6	3	80	129	0.11	0.17
	7	4	94	140	0.16	121.33
71	5	3	68	124	0.07	0.02
	6	3	80	133	0.12	0.12
	7	4	94	144	0.18	2.06
73	5	3	68	126	0.08	0.02
	6	3	80	135	0.12	0.11
	7	4	94	146	0.18	1.47
79	5	3	68	132	0.08	0.02
	6	4	80	141	0.12	0.07
	7	4	94	152	0.19	0.62
83	5	3	68	136	0.08	0.02
	6	4	80	145	0.13	0.04
	7	3	94	156	0.21	0.29
89	5	3	68	142	0.09	0.02
	6	3	80	151	0.14	0.03
	7	3	94	162	0.21	0.17
97	5	3	68	150	0.09	0.02
	6	3	80	159	0.14	0.03
	7	4	94	170	0.22	0.10

Experiment 2						
n	ℓ	d_{reg}	#Var	$\#P_{\text{equ}}$	T_{Inter}	T_{GB}
37	5	4	62	84	0.04	0.04
	6	4	76	95	0.06	4.23
	7	4	90	106	0.09	27.87
41	5	4	62	88	0.03	0.04
	6	4	76	99	0.06	0.49
	7	4	90	110	0.09	11.45
43	5	3	62	90	0.04	0.05
	6	4	76	101	0.06	5.35
	7	4	90	112	0.10	15.360
47	5	4	62	94	0.04	0.06
	6	4	76	105	0.06	1.28
	7	4	90	116	0.13	8.04
53	5	3	62	100	0.04	0.02
	6	4	76	111	0.06	0.19
	7	4	90	122	0.14	68.23
59	5	4	62	106	0.04	0.02
	6	3	76	117	0.07	0.11
	7	4	90	128	0.11	4.34
61	5	4	62	108	0.04	0.02
	6	3	76	119	0.07	0.09
	7	4	90	130	0.11	5.58
67	5	4	62	114	0.04	0.02
	6	4	76	125	0.07	0.07
	7	4	90	136	0.11	0.94
71	5	4	62	118	0.04	0.02
	6	4	76	129	0.07	0.04
	7	3	90	140	0.12	0.25
73	5	4	62	120	0.05	0.02
	6	4	76	131	0.07	0.03
	7	3	90	142	0.13	0.22
79	5	4	62	126	0.05	0.02
	6	4	76	137	0.08	0.03
	7	4	90	148	0.12	0.33
83	5	4	62	130	0.05	0.02
	6	4	76	141	0.09	0.03
	7	4	90	152	0.13	0.13
89	5	4	62	136	0.05	0.02
	6	4	76	147	0.09	0.03
	7	4	90	158	0.13	0.05
97	5	4	62	144	0.05	0.02
	6	4	76	155	0.09	0.03
	7	4	90	166	0.13	0.04

Table 3.4: Comparison of solving our systems of equations, having a solution, using Gröbner basis methods in Experiment 1 and Experiment 2 for $m = 4$. Notation is as above. The second tabular column already appeared in Table 3.1.

Experiment 1						
n	ℓ	d_{reg}	#Var	$\#P_{\text{equ}}$	T_{Inter}	T_{GB}
17	3	5	36	41	590.11	216.07
	4	*	*	*	*	*
19	3	5	36	43	564.92	211.58
	4	*	*	*	*	*
23	3	5	36	47	1080.14	146.65
	4	*	*	*	*	*
29	3	5	36	53	1069.49	232.49
	4	*	*	*	*	*
31	3	5	36	55	837.77	118.11
	4	*	*	*	*	*
37	3	5	36	61	929.82	178.04
	4	*	*	*	*	*
41	3	4	36	65	1261.72	217.22
	4	*	*	*	*	*
43	3	4	36	67	1193.13	220.25
	4	*	*	*	*	*
47	3	4	36	71	1163.94	247.78
	4	*	*	*	*	*
53	3	4	36	77	1031.93	232.110
	4	*	*	*	*	*

Experiment 2						
n	ℓ	d_{reg}	#Var	$\#P_{\text{equ}}$	T_{Inter}	T_{GB}
17	3	4	54	59	0.29	0.29
	4	4	67	68	0.92	51.79
19	3	4	54	61	0.33	0.39
	4	4	71	74	1.53	33.96
23	3	4	54	65	0.26	0.31
	4	4	75	82	2.52	27.97
29	3	4	54	71	0.44	0.50
	4	4	77	90	3.19	35.04
31	3	4	54	73	0.44	0.58
	4	4	77	92	3.24	9.03
37	3	4	54	79	0.36	0.43
	4	4	77	98	3.34	9.07
41	3	4	54	83	0.40	0.54
	4	4	77	102	3.39	17.19
43	3	4	54	85	0.43	0.53
	4	4	77	104	3.44	9.09
47	3	4	54	89	0.50	0.65
	4	4	77	108	3.47	9.59
53	3	4	54	95	0.33	0.40
	4	4	77	114	11.43	11.64

3.7 Splitting method to solve DLP for binary curves

Despite making use of symmetries to speed up the point decomposition problem, our experiment to solve the DLP for binary Edwards curve using the index calculus algorithm are still limited for small parameters n, ℓ, and m. This is mainly because the degree of the summation polynomial is high and when the Weil descent is made, the number of variables over the field $\mathbb{F}_2$ is quite large. We have concluded that cryptosystems making use of the DLP for binary curves as their security assumption are safe.

To lower the degree of the summation polynomials, a new idea is suggested in [Sem15, PTH15, Kar15, HKY15]. The idea is to split the summation polynomial into lower degree summation polynomials. The lowering of the degree of the summation polynomial comes at a cost of introducing new variables which in turn increases the complexity.

Let $\mathbb{K} = \mathbb{F}_{2^n}$ for a prime n. Let E be a binary elliptic curve defined over the field $\mathbb{K}$ given by the Weierstrass equation $E : y^2 + xy = x^3 + a_2x^2 + a_6$, where $\{a_2, a_6\} \in \mathbb{K}$. For the curve E, the 3^{rd} summation polynomial due to Semaev [Sem04] is given by

$$f_3(x_1, x_2, x_3) = x_1^2x_2^2 + x_1^2x_3^2 + x_2^2x_3^2 + x_1x_2x_3 + a_6. \tag{3.11}$$

For $m \geq 4$, the summation polynomials are constructed using resultants recursively as follows

$$f_m(x_1, \cdots, x_m) = \text{Resultant}_x(f_{m-j}(x_1, x_2, \cdots, x_{m-j-1}, x), f_{j+2}(x_{m-j}, x_{m-j+1}, \ldots, x_m, x))$$

for $1 \leq j \leq m - 3$.

Let the factor base be defined in the usual way as

$$\mathcal{F} = \{(x, y) \in E(\mathbb{F}_{2^n}) \mid x \in V\}, \tag{3.12}$$

where V is a vector subspace of $\mathbb{F}_{2^n}$ of dimension ℓ such that $n \approx \ell m$. Let $(1, \theta, \theta^2, \cdots, \theta^{\ell-1})$ be the basis of V and $(1, \theta, \theta^2, \cdots, \theta^{n-1})$ be the basis of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$, where θ is a root of a degree n irreducible polynomial f with coefficients in $\mathbb{F}_2$.

The splitting idea [Sem15, PTH15, Kar15, HKY15] is based on the observation that decomposing a random element R as a sum of m factor base elements P_i is equivalent to decomposing R as a sum of intermediate subset sums of these m factor base elements P_i. So we have $R = P_1 + P_2 + \cdots + P_m$ if and only if the possible intermediate sums of these points sum to R. For example if $m = 3$, we have $R = P_1 + P_2 + P_3$ if and only if $R = R_1 + P_3$, where $R_1 = P_1 + P_2$. If $m = 4$, we have $R = P_1 + P_2 + P_3 + P_4$ if and only if

$$R = \begin{cases} R_1 + R_2, & \text{where } R_1 = P_1 + P_2, \text{ and } R_2 = P_3 + P_4 \\ P_4 + R_3, & \text{where } R_1 = P_1 + P_2, \text{ and } R_3 = P_3 + R_1 \\ R_1 + P_3 + P_4, & \text{where } R_1 = P_1 + P_2. \end{cases} \tag{3.13}$$

Note that $R_1, R_2 \in E(\mathbb{F}_{2^n})$. The summation polynomials corresponding to these decompositions respectively are

$$f_5(x_1, x_2, x_3, x_4, x(R)) = 0 \iff \begin{cases} f_3(x(R_1), x(R_2), x(R)) = 0 \\ f_3(x_1, x_2, x(R_1)) = 0 \\ f_3(x_3, x_4, x(R_2)) = 0, \end{cases} \tag{3.14}$$

$$f_5(x_1, x_2, x_3, x_4, x(R)) = 0 \iff \begin{cases} f_3(x(R_3), x_4, x(R)) = 0 \\ f_3(x_1, x_2, x(R_1)) = 0 \\ f_3(x_3, x(R_1), x(R_3)) = 0, \end{cases} \tag{3.15}$$

$$f_5(x_1, x_2, x_3, x_4, x(R)) = 0 \iff \begin{cases} f_3(x_1, x_2, x(R_1)) = 0 \\ f_4(x_3, x_4, x(R_1), x(R)) = 0. \end{cases} \tag{3.16}$$

The original summation polynomial $f_5(x_1, x_2, x_3, x_4, x(R))$ is of degree 8 in each of the 4 variables. On the other hand the summation polynomials on the right side of equation (3.14) and (3.15) are of degree 2 in each of the 3 variables. So instead of applying Weil descent on f_5, the Weil descent is applied on these derived 3^{rd} summation polynomials.

From our trivial observation, the degree of the derived summation polynomials is lower than the original summation polynomial. So we expect a speed up in solving the corresponding polynomial systems. However the number of intermediate field variables ("auxiliary variables") introduced in turn increase the complexity of solving the polynomial systems. For example the system of polynomial equations derived from the summation polynomials on the right side of equation (3.14) or (3.15) introduce the variables $x(R_1), x(R_2) \in \mathbb{F}_{2^n}$ and hence an increase by $2n$ additional binary variables in the Weil descent of these polynomial systems.

There is a typical tradeoff between lowering the degree of the summation polynomials and increasing the number of variables. As in previous discussions, the two important parameters that determine the complexity of solving such polynomial systems using the F4 or F5 Gröbner basis algorithms are the degree of regularity and the number of variables involved in the system.

The independent contributions from [Kar15, Sem15, PTH15] are almost similar (see also [HKY15]). For example in [Kar15], the original summation polynomial is split into two derived summation polynomials as shown by equation (3.16). Whereas in [Sem15] and [PTH15], the summation polynomial is split into $m - 1$ derived summation polynomials as shown in equation (3.14). We focus on the latter. So if $R = P_1 + P_2 + \cdots + P_m$ then

$$f_{m+1}(x_1, \cdots, x_m, x(R)) = 0 \iff \begin{cases} f_3(x_1, x_2, x(R_1)) = 0 \\ f_3(x_3, x(R_1), x(R_2)) = 0 \\ f_3(x_4, x(R_2), x(R_3)) = 0 \\ \vdots \\ f_3(x_m, x(R_{m-2}), x(R)) = 0. \end{cases} \tag{3.17}$$

We apply Weil descent on the right hand side of equation (3.17) instead of applying the Weil descent on the summation polynomial f_{m+1}. This introduces $m-2$ intermediate field variables of the form $x(R_i)$. After the Weil descent, we have $n(m-2)$ more binary variables than the original system.

Consider the 3^{rd} summation polynomial $f_3(x_1, x_2, x(R_1)) = 0$. The first fall degree d_{ff} of the polynomial system F obtained by applying Weil descent to the 3^{rd} summation polynomial is 3 (see [Kar15]). Indeed the monomials we get after the Weil descent are

$$\{1, x_1^2x_2^2, x_1^2x_R^2, x_2^2x_R^2, x_1x_2x_R\},$$

where x_R is $x(R)$. So for each polynomial $f_i \in F$, we have $\max_i(\deg(f_i)) = 3$ over $\mathbb{F}_2$. By the definition of the first fall degree (see Definition 2.1.25), taking $g = x_1$, we have $\max_i(\deg(f_i) + \deg(g)) = 4$ but $\deg(x_1 f_i) = 3$. Note that the monomials of x_1F are

$$\{x_1, x_1^3x_2^2, x_1^3x_R^2, x_1x_2^2x_R^2, x_1^2x_2x_R\}.$$

Since $x_i^2 - x_i = 0$ over $\mathbb{F}_2$, we have $\deg(x_1 f_i) = 3$.

Under the assumption that the degree of regularity $d_{\text{reg}} \leq d_{\text{ff}} + 1$, we expect $d_{\text{reg}} \leq 4$. Indeed experiments made by the three independent contributions show that this is the case for small parameters. The experiments made are not conclusive. In [PTH15], experiments for the case $m \in \{3, 4, 5, \}$ and small parameters n, ℓ show that the degree of regularity almost stayed at 4 which is consistent with the first fall degree assumption.

The splitting technique gives an advantage over existing approach. For example in our experiment, we handled decompositions up to $m = 4$ for small parameters n, ℓ (see Table 3.4). Whereas experiments made in [PTH15] handle up to $m = 6$ although the running time is huge. The best running times recorded are 122559.83, 48122.94, and 88161.21 seconds, which is approximately 34, 13, and 24 hours for the (n, ℓ, m) pairs $(29, 3, 4)$, $(29, 3, 5)$, and $(23, 3, 6)$ respectively.

New factor base definition

The probability of decomposing a random element R as a sum of m elements of the factor base is $1/(m!)$. If m is large, we require fewer relations. Thus the relation search stage and linear algebra costs are reduced. However the probability of finding a relation is decreased. We give one possible definition of the factor base to bring the probability of finding a relation close to 1. This is a similar technique to the factor base definition we saw in Chapter 3 Section 3.4.

Definition 3.7.1. *(Factor base)*

$$\mathcal{F}_i \;=\; (x, y) \in E \;\mid\; x \in \mathit{span}\{\theta^{(i-1)\ell}, \theta^{(i-1)\ell+1}, \cdots, \theta^{i\ell-1}\} \;\; \mathit{for}\; 1 \leq i \leq m.$$

Note that $\mathcal{F}_1 \cap \mathcal{F}_2 \cap \cdots \cap \mathcal{F}_m = 0$. Heuristically, the size of each factor base $\mathcal{F}_i$ is 2^ℓ. Under the above definition of the factor base, the probability of decomposing a random element as a sum of m factor base elements is $2^{\ell m}/2^n \approx 1$.

Chapter 4

The DLP for Supersingular Ternary Curves

Contents

In this chapter we consider the question of whether an automorphism of an elliptic curve can be used to speed up the point decomposition problem. The previous chapter has used the automorphism [-1] as well as the translation map by a point of order 2 or 4. As a first case of study we consider an automorphism of order 3 arising from a supersingular curve. The point of this section is not to attack supersingular curves, but to attack curves with non-trivial automorphisms. Our preliminary findings are that, unlike low order torsion points, general automorphisms of elliptic curves do not seem to be a useful tool to speed-up the actual point decomposition problem. Indeed, our approach is worse than just forgetting the automorphism. On the positive side, our approach does reduce the size of the factor base. That means the number of relations required in the relation search stage is reduced. So in an indirect way, it gives a speed up in the point decomposition problem. It also reduces the cost of the linear algebra.

4.1 Elliptic curve over a field of characteristic three

Recall that elliptic curves can be divided into ordinary and supersingular elliptic curves. Supersingular curves defined over $\mathbb{F}_{3^n}$ or $\mathbb{F}_{2^n}$ are special types of elliptic curves which are very important for the implementation of pairing based cryptosystems. These curves have small embedding degree. Thus the Tate or Weil pairing can be computed efficiently using Miller's algorithm.

We also observe the existence of an automorphism of order 3, whose action is simple (not complicated), in supersingular elliptic curves defined over $\mathbb{F}_{3^n}$. Thus it is worth investigating the DLP for these curves.

Consider the Weierstrass equation E given by

$$E : y^2 = x^3 - x + 1$$

to be the supersingular elliptic curve defined over $\mathbb{F}_{3^n}$. We recall the group law for point addition and point doubling adapted for this curve. Let $P_1 = (x_1, y_1), P_2 = (x_2, y_2) \in E$. The group law for point addition and doubling is given as follows.

1. **(Point Addition:)** Assume $P_1 \neq \pm P_2$ and let λ be the slope joining P_1 and P_2, $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$, then $P_1 + P_2 = (x_3, y_3)$ where,

$$x_3 = \lambda^2 - x_1 - x_2 \quad \text{and} \quad y_3 = (y_1 + y_2) - \lambda^3.$$

2. **(Point Doubling):** Let $P_1 \neq -P_1$ and $\lambda = \frac{1}{y_1}$ be the slope of the line through P_1 which is tangent to the curve E, then $P_1 + P_1 = (x_3, y_3)$ where,

$$x_3 = \lambda + x_1 \quad \text{and} \quad y_3 = -(y_1 + \lambda^3).$$

Following the work of Semaev [Sem04], one can easily compute the 3^{rd} summation polynomial to be

$$\begin{aligned} f_3(x_1, x_2, x_3) \quad = \quad & (x_1^2 + x_2^2 + x_1 x_2)x_3^2 + (x_1^2 x_2 + x_1 x_2^2 - x_1 - x_2 - 1)x_3 + \\ & x_1^2 x_2^2 - x_1 x_2 - x_1 - x_2 + 1. \end{aligned} \tag{4.1}$$

This corresponds to decomposing a random point $R = (x_3, y_3)$ as a sum of two elements of a factor base, $R = P_1 + P_2$, where $P_1 = (x_1, y_1)$, and $P_2 = (x_2, y_2)$. The factor base is defined in the usual way as a vector subspace of $\mathbb{F}_{3^n}$ of dimension ℓ.

As discussed in previous chapters, the summation polynomials of higher degrees are constructed recursively. By construction, the summation polynomials are symmetric. So they are invariant under the action of the symmetric group S_m. We can then re-write the summation polynomials using the primary symmetric invariant polynomials in the variables x_i. This lowers the degree by $m!$, the order of the group. For example, the 3^{rd} summation polynomial f_3 given in equation (4.1) can be re-written as

$$\tilde{f}_3(e_1, e_2, x_3) = (e_1^2 - e_2)x_3^2 + (e_2 e_1 - e_1 - 1)x_3 + e_2^2 - e_2 - e_1 + 1, \tag{4.2}$$

where $e_2 = x_1 x_2$ and $e_1 = x_1 + x_2$ are the fundamental symmetric invariant variables. Clearly solving equation (4.2) is easier than solving equation (4.1) after both are evaluated at x_3 .

Assume we have a larger group acting on the summation polynomials such that the action leaves them invariant. Then one can imagine that if the generators of the invariant ring under the group can easily be computed, then by re-writing the original summation polynomial in terms of the new generators we get a lowered degree. In the following section we will examine an automorphism group whose action leaves the summation polynomials invariant.

4.2 Automorphisms and resolution of point decomposition problem

The Pollard rho/lambda algorithm and its variants are currently the most effective generic algorithms to solve the DLP problem in elliptic curves. As discussed in 2.2.2, an automorphism of order k of an elliptic curve is used to speed up the Pollard rho/lambda algorithm by $\sqrt{k}$ (see [DGM99]). The overall expected running time of the algorithm is $\tilde{O}(\sqrt{\pi n/2}/(\sqrt{k}M))$ group operations, where M is the number of processors used and n is the group order.

The index calculus algorithm solves the DLP defined in hyperelliptic curves in subexponential time. The presence of an order k automorphism speeds up the algorithm. Specifically it decreases the factor base by a factor of k. So the number of relations required is decreased by the same quantity which in turn speeds up the running time of the linear algebra by a factor of k^2. We refer to [Gau09] for details.

The research on DLP defined in elliptic curves seems to focus on exploiting low order points, whose group action is simple (not complicated), to speed up the PDP. To explore other ways, we investigate the presence of an automorphism group of order 3, whose action is simple under the group law operation, in elliptic curves defined over a field of characteristic 3.

Let $P = (x, y) \in E$. Consider the Frobenius map $\phi : (x, y) \mapsto (x^3, y^3)$. Let ψ be $\phi + 1$. Then ψ maps:

$$(x, y) \mapsto (x + 1, y).$$

Indeed $\psi(P) = (\phi + 1)(P) = P + \phi(P)$. It can be easily verified using the group law that $(x, y) + (x^3, y^3) = (x + 1, y)$. So $\psi \in End(E)$ satisfying the characteristic polynomial $x^2 - 5x + 7$, where $End(E)$ refers the endomorphism ring of E. Clearly ψ has order three. In [Kob98], it is denoted as ω and it is used for efficient supersingular implementation of the elliptic curve digital signature algorithm in a field of characteristic three.

Assume a random element $R \in E$ can be written as sum of m elements of the factor base, that is $R = P_1 + P_2 + \cdots + P_m$. Then the action of ψ on the decomposition of R produces decompositions of the form

$$\psi^j(R) = \psi^j(P_1) + \psi^j(P_2) + \cdots + \psi^j(P_m)$$

for $1 \leq j \leq 3$. If $(x_1, x_2, \cdots, x_m, x(R))$ is a solution to the corresponding summation polynomial, then the solution $(x_1 + j, x_2 + j, \cdots, x_m + j, x(R) + j)$ corresponds to the action of ψ^j for $1 \leq j \leq 3$.

Let $G = H \times S_m$ be a group which is a product of the groups $H = \mathbb{Z}_3$ and S_m, where $H = \mathbb{Z}_3$ is a group of order 3 representing the automorphism group and S_m is the symmetric group with order $m!$. The action of the group G on the summation polynomial is a combination of addition of j for $j \in \{0, 1, 2\}$ to each variable simultaneously accompanied by a permutation of the variables. Clearly the action leaves the summation polynomial invariant. So if one can find the generators of the invariant ring under G, one can hope that the degree of the summation polynomial could be reduced by the order of G, that is by a factor of $3m!$.

To exploit the action of the automorphism group, we define the factor base in such a way that if $(x, y) \in E$ is in the factor base, then the points $(x + 1, y), (x + 2, y)$ are also in the factor base. By keeping only one element in the factor base for each orbit under ψ^j for $1 \leq j \leq 3$, the size of the factor base is reduced by a factor of 3. This reduces the number of relations required by a factor of 3 and the linear algebra cost by a factor of 9. We also note that the trivial automorphism which sends $(x, y) \mapsto (x, -y)$ reduces the factor base by a factor of 2. Hence in total the number of relations requirement is reduced by a factor 6 and the linear algebra cost is reduced by a factor of 36.

4.3 Invariant rings under the automorphism and symmetric groups

Let $R = P_1 + P_2 + \cdots + P_m$ be a relation. Unlike the action of low order torsion points, the action of H on this relation is both on the sum R and the points of the factor base summing to R. In other words, we get another valid relation for a different element Q which is related to R. Note that in the case of the action of low order torsion points for the same R we get different relations.

Let $m = 2$ be fixed. Then the action of G on $f_3(x_1, x_2, x_3)$ is a combination of the action of $H = \mathbb{Z}_3$ and the symmetric group S_2. We introduce a "dummy" variable y which is fixed to be 1 to homogenize the action of H. We order the variables as y, x_1, x_2, x_3, where the x_i are the indeterminate variables. Let H and S_2 be generated by the matrices $\mathbf{A}$ and $\mathbf{B}$ respectively. Then

$$\mathbf{A}\begin{pmatrix} y \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y \\ x_1 + y \\ x_2 + y \\ x_3 + y \end{pmatrix} \quad \text{and} \quad \mathbf{B}\begin{pmatrix} y \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y \\ x_2 \\ x_1 \\ x_3 \end{pmatrix}.$$

So the matrices $\mathbf{A}$ and $\mathbf{B}$ are represented by

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

We need to find the generators of the invariant ring under the group G. Since the characteristic of $\mathbb{K}$ divides the order of G ($3 \mid \#G$), finding the generators of the invariant ring $\mathbb{K}[x_1, x_2, \cdots, x_n]^G$ is cumbersome. So we will adopt an *ad hoc* technique to find the generators of an invariant ring under G using the Singular computer algebra system [DGPS15]. The system uses Kemper's algorithm [Kem96] for "calculating invariant rings of finite groups over arbitrary fields". The algorithm makes use of the following proposition.

Proposition 4.3.1. *(Kemper [Kem96]) Let $\tilde{G}$ be a finite group. Let $F = \{f_1, \cdots, f_i\} \subset \mathbb{K}[x_1, \cdots, x_n]^{\tilde{G}}$ be a set of homogeneous invariants. Then F can be extended to a system of primary invariants if and only if*

$$\dim(f_1, \cdots, f_i) = n - i.$$

Moreover the set F forms a system of primary invariants if and only if $i = n$ and $V(F) = \{0\}$, where $V(F)$ denotes the variety of F over the algebraic closure of $\mathbb{K}$.

Kemper's algorithm (see [Kem96]) starts with low degree polynomials to find polynomial basis $f_1, \cdots, f_i$ fulfilling Proposition 4.3.1. A base element f_{i+1} is added to this set if it does not lie in the radical of the ideal spanned by the f_i.

We provide a Singular source code to compute the primary invariants for the invariant ring under $G = \mathbb{Z}_3 \times S_2$ for the case $m = 2$ which can be extended for larger m.

Line 1: LIB "finvar.lib"

Line 2: ring $R = 3, (y, x_1, x_2, x_3), dp$;

Line 3: matrix $A[4][4] = 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1$;

Line 4: matrix $B[4][4] = 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1$;

Line 5: matrix $D(1..2) =$ invariant_ring(A, B);

Line 6: $D(1)$;

Executing the above program, we get the primary invariants, denoted as s_i, of the invariant ring under the group G to be

$$\begin{aligned} s_1 &= x_1 + x_2 + x_3, \\ s_2 &= x_1^2 + x_1 x_2 + x_2^2, \\ s_3 &= x_1 - x_1^3 + x_2 - x_2^3. \end{aligned}$$

So now we know the primary invariant of the invariant ring under the group $G = \mathbb{Z}_3 \times S_2$. This is sufficient to test if these invariants actually speed up the point decomposition problem. Consider the 3^{rd} summation polynomials $f_3(x_1, x_2, x_3)$ corresponding to the decomposition $P_1 + P_2 = R$ with x_3 is the known x-coordinate of the random point of R and $x_i = x(P_i)$. In the standard approach, first we evaluate the summation polynomial at x_3 and then we re-write it using the invariant variables. Finally we apply Weil descent. But here due to the action of H (x_3 is not fixed), we re-write f_3 using the invariant variables then we evaluate and apply the Weil descent respectively. One can observe that the invariant variable S_1 no more lies in our chosen subspace for the factor base but instead lies in $\mathbb{F}_{3^n}$ (observe that x_3 lies in $\mathbb{F}_{3^n}$). As a result, the number of variables in the resulting system is increased and the resulting system is relatively harder to solve which cannot be compensated due to the group action. This is also verified in our experiment which is not reported here.

We conclude that an automorphism of an elliptic curve can speed up the point decomposition problem by reducing the factor base. As a result we require fewer relations in the relation search stage and the cost of the linear algebra is reduced. But an automorphism of an elliptic curve is worse in the actual polynomial system solving.

The MOV attack (see section 2.2.3), which transfer the DLP to finite fields, remain the best attack against supersingular ternary curves. Currently the DLP record for characteristic three finite field is $\mathbb{F}_{3^{5\times 479}}$ (see [JP14]).

Chapter 5

The Approximate Common Divisor Problem and Lattices

Contents

The security of the DGHV cryptosystem [DGHV10] and its variants such as [CMNT11] depend on the hardness assumption of the ACD problem (see Definition 5.2.1). In this chapter we focus on the application of lattices to solve the ACD problem. We review and compare existing algorithms to solve the ACD problem. Our main contribution is to simplify two standard lattice attacks on ACD and give a precise analysis of them. Under the assumption that a short basis for a given lattice exists, we show that we can recover a solution to the ACD problem from the kernel of a system of equations. This system is obtained from the short basis of the lattice using two existing algorithms based on the idea of orthogonal lattices. Another contribution is to compare the Cohn-Heninger approach with other methods. We find that the Cohn-Heninger algorithm is slower than other methods. Our third contribution is on preprocessing of ACD instances to extend the range of lattice attacks.

5.1 Lattices and computational assumptions

Lattices have many cryptographic applications. Similar to the design of cryptosystems based on the assumption that integer factoring and discrete logarithm problem are hard, we can design cryptographic primitives whose security relies on hard computational problems in lattices [AR04, Kho04]. In fact lattice based cryptographic schemes [Ajt96] and subsequent works [GGH97, Reg04] have advantages over existing systems. Existing cryptosystems such as *RSA* are based on average-case assumptions contrary to worst-case hardness assumed by lattice problems. This gives an extra confidence on the security of our cryptographic primitives: a proof based on the assumption that if one is able to break the cryptographic primitive with small probability then any instance of the lattice problem can be solved.

Another advantage of lattice based constructions is their efficiency. The arithmetic operations involved with lattices are only modular addition which is simple and can also be parallelized due to the structure of lattices. They are also currently the most promising alternatives to existing number theoretic based cryptographic schemes. If quantum computers are publicly available, such existing systems are not secure any more. An efficient quantum algorithm [Sho97] for factoring and solving discrete logarithm problems already exists. To date, there is no such attack against lattice computational problems.

Apart from cryptographic primitive constructions, lattices have been used as cryptanalytic tools as in [JS98], finding small roots in Copersmith's algorithms [Cop96a, Cop96b, Cop97] (see [NS01] for a survey on cryptanalytic application of lattices). For our application, we focus on the use of lattices to solve the approximate common divisor problem which is the heart of the security proof of the homomorphic encryption based on the integers [DGHV10]. So in this section we study lattices and some of the hard computational problems associated with lattices. We also introduce the use of lattices to solve the Knapsack problem which is quite similar to the way lattices are used to solve the approximate common divisor problem using orthogonal lattice based approach.

Lattice

Some references on lattices use columns and others use rows. The two theories are exactly the same. Here we use them interchangeably.

Definition 5.1.1. *(Lattice) A lattice is a set*

$$L = \{\mathbf{B}c \mid c \in \mathbb{Z}^n\} = \left\{\sum_i c_i.\mathbf{b}_i \mid c_i \in \mathbb{Z}\right\},$$

where $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n]$ *is a matrix of* n *linearly independent column vectors* $\mathbf{b}_i \in \mathbb{R}^m$.

The matrix $\mathbf{B}$ is called the basis of the lattice L. The parameters m, n refer to the dimension and rank of the lattice respectively. If $n = m$, the lattice is called full rank.

Definition 5.1.2. *(Fundamental Parallelepiped) Given a lattice basis matrix* $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n]$, *the fundamental parallelepiped associated with the matrix* $\mathbf{B}$ *is the set of points*

$$\mathcal{P}(\mathbf{B}) = \left\{\sum_i c_i.\mathbf{b}_i \mid 0 \le c_i < 1\right\}.$$

The determinant of the lattice L is given by the n-dimensional volume of the fundamental parallelepiped $\mathcal{P}(\mathbf{B})$. It can be shown that

$$\det(L) = \sqrt{\mathbf{B}^\top \mathbf{B}}, \tag{5.1}$$

where $\mathbf{B}^\top$ is the transpose of $\mathbf{B}$ (when using row lattices the formula is $\sqrt{\mathbf{B}\mathbf{B}^\top}$). If the lattice is full rank, the determinant of L is the same as the determinant of $\mathbf{B}$.

Definition 5.1.3. *(Gram Schmidt orthogonalization)*

Given a sequence of n linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n$, the Gram Schmidt orthogonalization is the sequence of vectors $\tilde{\mathbf{b}}_1, \tilde{\mathbf{b}}_2, \cdots, \tilde{\mathbf{b}}_n$ defined by

$$\tilde{\mathbf{b}}_i = \mathbf{b}_i - \sum_{j=i}^{i-1} \mu_{i,j} \tilde{\mathbf{b}}_j,$$

where $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle}{\langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{b}}_j \rangle}$.

Let $\tilde{\mathbf{B}} = [\tilde{\mathbf{b}}_1, \tilde{\mathbf{b}}_2, \cdots, \tilde{\mathbf{b}}_n]$. The lattice basis $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n]$ can be uniquely re-written as follows

$$\mathbf{B} = \tilde{\mathbf{B}} \begin{pmatrix} 1 & \mu_{2,1} & \mu_{3,1} & \cdots & \mu_{n,1} \\ & 1 & & \cdots & \mu_{n,2} \\ & & & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix} = \mathbf{C} \begin{pmatrix} ||\tilde{\mathbf{b}}_1|| & \mu_{2,1}||\tilde{\mathbf{b}}_1|| & \mu_{3,1}||\tilde{\mathbf{b}}_1|| & \cdots & \mu_{n,1}||\tilde{\mathbf{b}}_1|| \\ & ||\tilde{\mathbf{b}}_2|| & & \cdots & \mu_{n,2}||\tilde{\mathbf{b}}_2|| \\ & & & & \\ & & & \ddots & \\ & & & & ||\tilde{\mathbf{b}}_n|| \end{pmatrix},$$

where $\mathbf{C} = [\tilde{\mathbf{b}}_1/||\tilde{\mathbf{b}}_1||, \tilde{\mathbf{b}}_2/||\tilde{\mathbf{b}}_2||, \cdots, \tilde{\mathbf{b}}_n/||\tilde{\mathbf{b}}_n||]$ is the orthonormal basis obtained by normalization of the Gram Schmidt vectors $\tilde{\mathbf{b}}_1, \tilde{\mathbf{b}}_2, \cdots, \tilde{\mathbf{b}}_n$. The notation $||\mathbf{x}||$ denotes the ℓ_2 norm of the vector $\mathbf{x}$.

Lemma 5.1.4. *Let $\mathbf{B}$ be the basis matrix of the lattice L as above, then the determinant of L is given by $\prod_{i=1}^{n} ||\tilde{\mathbf{b}}_i||$.*

The length of the shortest non-zero vector, denoted by λ_1, is a very important parameter of a lattice. It is expressed as the radius of the smallest zero-centered ball containing a non-zero linearly independent lattice vector. More generally we have successive minima defined as follows:

Definition 5.1.5. (Successive minima) Let L be a lattice of rank n and dimension m. For $1 \leq i \leq n$, the i^{th}-successive minima is defined as

$$\lambda_i(L) = \min\{r \mid \dim(\text{span}(L \cap \mathcal{B}(0, r))) \geq i\},$$

where $\mathcal{B}(0, r) = \{x \in \mathbb{R}^m \mid ||x|| \leq r\}$ is a ball centered at the origin.

The number of lattice points within a ball is estimated by the Gaussian heuristic.

Heuristic 5.1.6. *(Gaussian Heuristic) Let L be a full rank lattice in $\mathbb{R}^m$. Let $\mathcal{K}$ be a measurable subset of $\mathbb{R}^m$, then the Gaussian heuristic estimates the number of lattice points in $\mathcal{K} \cap L$ to be approximately $\text{vol}(\mathcal{K})/\det(\text{L})$, where $\text{vol}(\mathcal{K})$ denotes the m dimensional volume.*

Heuristic 5.1.7. *(First Minima) Let L be a full rank lattice in $\mathbb{R}^m$. Then by the Gaussian heuristic the length of the shortest non-zero vector λ_1 in L is estimated by*

$$\lambda_1 = \sqrt{\frac{m}{2\pi e}} \det(L)^{1/m}. \tag{5.2}$$

Computational assumptions in lattices

There are some hard computational problems in lattices. The most well-known include the *Shortest Vector Problem (SVP)* and the Closest Vector Problem (CVP) along with their approximation version SVP$_\gamma$ and CVP$_\gamma$ respectively for some parameter $\gamma > 1$. We define them as follows.

Definition 5.1.8. *Let $\mathbf{B}$ be the basis matrix for a lattice L and λ_1 be the first minima. Let $\gamma > 1$ be a parameter. The Shortest Vector Problem (SVP) is to find a non-zero vector $\mathbf{v} \in L$ such that $||\mathbf{v}||$ is minimal (that is $||\mathbf{v}|| = \lambda_1$). The Approximate Shortest Vector Problem (SVP$_\gamma$) is to find a non-zero vector $\mathbf{v} \in L$ such that $||\mathbf{v}|| \leq \gamma\lambda_1$.*

Definition 5.1.9. *Let $\mathbf{B}$ be the basis matrix for a lattice L. Let $\gamma > 1$ be a parameter. Then the Closest Vector Problem (CVP) is given a target vector $\mathbf{w} \in \mathbb{R}^n$ to find a vector $\mathbf{v} \in L$ such that $||\mathbf{w} - \mathbf{v}||$ is minimal. The Approximate Closest Vector Problem (SVP$_\gamma$) is to find a vector $\mathbf{v} \in L$ such that $||\mathbf{w} - \mathbf{v}|| \leq \gamma||\mathbf{w} - \mathbf{Bx}||$ for all $\mathbf{x} \in \mathbb{Z}^n$.*

5.1.1 Algorithms to solve CVP and SVP

The SVP and CVP are hard problems. We refer to [Ajt98, DKS98, Mic98] for details. We mention some of the common algorithms to solve these problems.

Babai's rounding technique

Given a basis matrix $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n]$ for a lattice L and a target vector $\mathbf{w} \in \mathbb{R}^n$, the CVP is to find a non-zero vector $\mathbf{v} \in L$ such that $||\mathbf{w} - \mathbf{v}||$ is minimal. The rounding technique by Babai (see Chapter 18 of [Gal12]) is one method to solve the CVP. It works best if the lattice basis vectors $\mathbf{b}_i$ are orthogonal or close to orthogonal. The idea is to write $\mathbf{w}$ as $\mathbf{w} = \sum_{i=1}^{n} l_i \mathbf{b}_i$, where $l_i \in \mathbb{R}$. In other words, we compute $\mathbf{B}^{-1}\mathbf{w}$ to get the vector $(l_1, l_2, \cdots, l_n)$. Then a solution to the CVP is

$$\mathbf{v} = \sum_{i=1}^{n} \lfloor l_i \rceil \mathbf{b}_i,$$

where $\lfloor l_i \rceil$ is the nearest integer to the real number l.

Embedding technique

Let L be a full rank lattice with basis matrix $[\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n]$. Let $\mathbf{w} \in \mathbb{R}^n$ be a target vector for the CVP. Another alternative to Babai's rounding technique to solve the CVP is to use the embedding technique (see Chapter 18 of [Gal12]). Let the solution to the CVP correspond to the integers $(l_1, l_2, \cdots, l_n)$ such that $\mathbf{w} \approx \sum_{i=1}^{n} l_i \mathbf{b}_i$. Then define $\mathbf{e}$ to be

$$\mathbf{e} = \mathbf{w} - \sum_{i=1}^{n} l_i \mathbf{b}_i.$$

The idea of the embedding technique is to construct a lattice $\tilde{L}$ that contains the short vector $\mathbf{e}$. We define the lattice $\tilde{L}$ to have a basis matrix $\tilde{\mathbf{B}}$ such that

$$\tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n & \mathbf{w} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

Let $\tilde{e} = \begin{bmatrix} \mathbf{e} \\ 1 \end{bmatrix}$. Then one observes that $\tilde{e}$ is in the lattice $\tilde{L}$. Indeed taking linear combination of the columns of $\tilde{\mathbf{B}}$ with coefficients $(-l_1, -l_2, \cdots, -l_n, 1)$ gives the vector $\tilde{e}$. If $\mathbf{w}$ is very close to a lattice point in $\tilde{L}$ then we would like $\tilde{e}$ to be the shortest vector in $\tilde{L}$. Then if one solves SVP in the lattice $\tilde{L}$ to get a vector $\tilde{e}$ then a solution $\mathbf{v}$ to CVP is given by

$$\mathbf{v} = \mathbf{w} - e.$$

The success of the algorithm depends on the size of the vector $\mathbf{e}$ compared to short vectors in the lattice L. Let λ_1 be the first minima of the lattice L. We require $||\tilde{e}|| \leq \lambda_1$ which implies $||\mathbf{e}||^2 + 1 \leq \lambda_1$. Hence the size of e should be less than λ_1. Note that this is still not guaranteed to succeed. For example for a small integer $\mu \neq \pm 1$, $\mu \begin{bmatrix} \mathbf{w} \\ 1 \end{bmatrix} - \sum_{i=1}^{n} l_i \begin{bmatrix} \mathbf{b}_i \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{e}' \\ \mu \end{bmatrix}$ is close to $\mu\mathbf{w}$ but not to $\mathbf{w}$.

LLL algorithm

Definition 5.1.10. *A basis* $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n]$ *is* δ-LLL reduced if the following conditions are satisfied.

1. For all $1 \leq i \leq n$ and $j < i$, $|\mu_{i,j}| \leq \frac{1}{2}$.
2. For all $1 \leq i < n, j < i$, $\delta||\tilde{\mathbf{b}}_i||^2 \leq ||\mu_{i+1,i}\tilde{\mathbf{b}}_i + \tilde{\mathbf{b}}_{i+1}||^2$.

Theorem 5.1.11. *(*LLL algorithm [LLL82]) Given a basis $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n]$ of a lattice $L \subseteq \mathbb{Z}^m$, the LLL algorithm outputs a δ-LLL reduced basis $\mathbf{B}' = [\mathbf{b}'_1, \mathbf{b}'_2, \cdots, \mathbf{b}'_d]$ in polynomial time given by

$$\tilde{O}(n^5 m \log^3(\max_j ||b_j||)), \tag{5.3}$$

where the maximum is taken over all elements of $b_j \in \mathbf{B}$. The δ-LLL reduced basis satisfies various bounds. For $\delta = 3/4$, we have

1. $||\mathbf{b}'_1|| \leq 2^{\frac{n-1}{2}}\lambda_1$.
2. $||\mathbf{b}'_1|| \leq 2^{\frac{n-1}{4}}\det(\mathbf{B})^{\frac{1}{n}}$, and $||\mathbf{b}'_n|| \leq 2^{\frac{n(n-1)}{4(n-d+1)}}\det(\mathbf{B})^{\frac{1}{n-d+1}}$. Thus the LLL algorithm solves SVP_γ for the approximation factor $\gamma = 2^{\frac{n}{2}}$.

The LLL algorithm [LLL82] is a very popular lattice reduction algorithm in cryptography mainly used in the cryptanalysis [MV10] of public key systems. The BKZ algorithm is a blockwise generalizations of the LLL algorithm (see [GHKN06, GN08b, Sch87, SE94]), introduced by Schnorr and Euchner [SE91, SE94]. It has better approximation factors especially in high dimensions, but the running time is not polynomial time. For comparison of the heuristic running time versus the quality of the output of various blockwise lattice reduction algorithms, we refer to [GN08a].

The LLL algorithm and its blockwise generalization output an approximation of a short vector. Consider a lattice L, then the LLL algorithm can efficiently compute an approximation of a short vector up to $2^{\frac{\dim(L)}{2}}$. So if $\mathbf{v}$ is the shortest lattice vector, then $\mathbf{v}$ can be found using the LLL algorithm if

$$\lambda_1 = ||\mathbf{v}|| < 2^{(\dim(L)-1)/2}\lambda_1(L) < \lambda_2(L).$$

On the other hand, block based lattice reduction algorithms approximately take time $2^{\dim(L)/k}$ to solve SVP_{2^k} [AKS01, Sch87].

Other lattice reduction algorithms include enumeration algorithms which are either space efficient [GNR10, FP85, Poh81] or with exponential space requirement [AKS01, MV10, NV08]. Unlike the approximation based lattice reduction algorithms, these types of lattice reduction algorithms output an exact short vector (with no approximation). They do not run in polynomial time and so are only useful in low dimensional examples.

For our application we will focus on the LLL algorithm.

5.1.2 Solving Knapsack problem

The Knapsack problem is given an integer b and a set $S = \{s_1, s_2, \cdots, s_t\}$ of positive integers, is there $P \subseteq S$ that sum to b. So, we have the following formal statement of the problem

Definition 5.1.12. *(Knapsack problem) Let $S = \{s_1, s_2, \cdots, s_t\}$, and b be positive integers. The Knapsack problem is to find $x_i \in \{0, 1\}$, if they exist, such that*

$$b = \sum_{i=1}^{t} x_i s_i.$$

Cryptosystems [MH78, LO85] make use of the Knapsack problem for proof of their security. One observes that finding an arbitrary solution to the linear equation is not hard; that is finding integer values $(y_1, y_2, \cdots, y_t)$ such that $\sum_{i=1}^{t} y_i s_i = b$ can be achieved in polynomial time using the extended euclidean algorithm. But if the y_i are required to belong to $\{0, 1\}$ the problem becomes hard.

Definition 5.1.13. *Given the set $S = \{s_1, s_2, \cdots, s_t\}$ of positive integers. The Knapsack density is defined as*

$$d = \frac{t}{\max_{1 \leq i \leq t}(\log_2(s_i))}.$$

If the Knapsack problem in consideration has low-density, the problem can be reduced to solving the short vector problem using the embedding technique. Let $\mathbf{x} = (x_1, \cdots, x_t)$, and $\mathbf{y} = (y_1, \cdots, y_t)$ be exact and arbitrary solutions of the Knapsack problem respectively. Let $\mathbf{z} = (y_1 - x_1, \cdots, y_t - x_t)$, then $\sum_{i=1}^{t} (y_i - x_i) s_i = 0$. We build a lattice L spanned by all integer vectors $\mathbf{z}$ such that $\sum_{i=1}^{t} z_i s_i = 0$. In other words, a vector $\mathbf{z} = (z_1, z_2, \cdots, z_t) \in L$ if and only if

$$z_1 s_1 + z_2 s_2 + \cdots + z_t s_t = 0.$$

The lattice L has dimension $t - 1$ with determinant

$$\det(L) = \sqrt{\sum_{i=1}^{t} s_i^2} \approx 2^{t/d}\sqrt{t}$$

[NS01], where d is the Knapsack density.

Let $\mathbf{B}$ be the basis of L. To solve the Knapsack problem, we construct a lattice of embedding L_1 spanned by the basis of the lattice L and the arbitrary integer solution $\mathbf{y}$. So the lattice L_1 is spanned by

$$\begin{pmatrix} \mathbf{B} \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \mathbf{y} \\ 1 \end{pmatrix}.$$

If $\mathbf{z} = (y_1 - x_1, \cdots, y_t - x_t) \in L$. We observe that $||\mathbf{z} - \mathbf{y}|| = ||\mathbf{x}|| \approx \sqrt{t/2}$. Then with high probability the vector $\mathbf{v} = (\mathbf{z} - \mathbf{y}) = (x_1, \cdots, x_t, 1)$ is the shortest vector in the lattice L_1 [NS01].

By the Gaussian heuristic the vector $\mathbf{v}$ is likely to be the shortest vector in L if

$$\sqrt{t/2} = ||\mathbf{v}|| < \det(L)^{1/t}\sqrt{\frac{t}{2\pi e}} \approx 2^{1/d}\sqrt{\frac{t}{2\pi e}} \implies 2^{1/d} > \sqrt{\pi e}.$$

We observe that $2^{1/d} > \sqrt{\pi e} \implies d < \frac{1}{\log_2(\sqrt{\pi e})} \approx 0.64$ [NS01, LO85]. The bound can be improved by taking $\mathbf{y} = (y_1 - 1/2, y_2 - 1/2, \cdots, y_n - 1/2)$ [NS01, CJL+92] instead of $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ in the lattice of embedding construction.

5.2 Algorithms to solve the approximate common divisor problem

Lattices are also used to solve the approximate common divisor problem, which is the heart of the homomorphic encryption of van Dijk et al. [DGHV10].

A homomorphic encryption scheme is an encryption scheme that allows computations (addition, and multiplications) on encrypted data without learning anything about the plain data. In other words if Encrypt$(m_i) = c_i$, then given $\tilde{f}$ there exists f such that

$$\text{Decrypt}(f(c_1, c_2, \cdots, c_t)) = \tilde{f}(m_1, m_2, \cdots, m_t),$$

where $\tilde{f}$ is an allowed boolean function on the original data. A major application area in cryptography is in cloud computing. As we know storage and processing power are critical and yet we are interested in computing with huge amounts of data sensitive data (say medical records). One possible option is to outsource our computations to a third party, say to a cloud. But we also like nothing to be learned about our data. So we encrypt our data and allow the cloud to make arithmetic operations on the encrypted data. Only the authorised user with the private key can decrypt the computed encrypted data corresponding to computations on our original data.

More formally, let λ be a security parameter. A homomorphic encryption scheme is composed of four probabilistic polynomial time algorithms

$$(\text{KeyGen}, \text{Encrypt}, \text{Decrypt}, \text{Evaluate})$$

each defined as follows.

Key Generation: $(pk, sk) \leftarrow \text{KeyGen}(\lambda)$. The polynomial time algorithm KeyGen(λ) generates public encryption key pk, and secret decryption key sk on input security parameter λ.

Encryption: $c_i \leftarrow \text{Encrypt}(pk, m_i)$. The encryption algorithm Encrypt(pk, m_i) takes input the public encryption key pk, and a plain text message $m_i \in \mathcal{M}$ and outputs a cipher text $c_i \in \mathcal{C}$, where $\mathcal{M}$ and $\mathcal{C}$ are the message and cipher text spaces respectively. If $m = (m_1, m_2, \cdots, m_t)$ and $c = (c_1, c_2, \cdots, c_t)$, we have $c \leftarrow \text{Encrypt}(pk, m)$.

Decryption: $m_i \leftarrow \text{Decrypt}(sk, c_i)$. On input cipher text c_i, and secret encryption sk, the decryption algorithm Decrypt(sk, c_i) outputs the plain text message m_i.

Evaluate: $c* \leftarrow \text{Evaluate}(pk, \tilde{f}, c_1, c_2, \cdots, c_t)$. The evaluation algorithm

$$\text{Evaluate}(pk, \tilde{f}, c_1, c_2, \cdots, c_t)$$

takes a public encryption key pk, a function $\tilde{f}$, where $\tilde{f}$ is an allowed boolean function on the original data, and a list of ciphertexts $(c_1, c_2, \cdots, c_t)$ that are encryption of $(m_1, m_2, \cdots, m_t)$ respectively as input. Then it computes a function f, where f is an allowed boolean function on the encrypted data and outputs a cipher text $c*$ using f which is a valid encryption of $\tilde{f}(m_1, m_2, \cdots, m_t)$ under the public encryption key pk.

The DGHV scheme [DGHV10] and other variants such as [CMNT11] have simple constructions; as they make use of the hardness of the partial approximate common divisor problem for their proof of security. There are four important security parameters that are to be appropriately set in the construction of the DGHV scheme. These are γ, η, and ρ which represent the bit length of integers in the public key, secret key and the noise respectively. The fourth parameter τ represents the number of integers in the public keys. These parameters are set to $(\lambda^5, \lambda^2, \lambda, \gamma + \lambda)$ respectively, where λ is a security parameter.

Define $[y]_x$ to be $y \pmod x$ and $\mathcal{D}_{\gamma,\rho}(p)$ to be the following efficiently sampleable distribution over γ bit integers for an η bit odd prime p.

$$\mathcal{D}_{\gamma,\rho}(p) = \{pq + r \mid q \leftarrow \mathbb{Z} \cap [0, 2^\gamma/p), r \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)\}. \tag{5.4}$$

Then the DGHV scheme is constructed as follows.

KeyGen(λ). On input security parameter λ, the key generation algorithm KeyGen(λ) generates an η bit odd prime integer p,

$$p \leftarrow (2\mathbb{Z} + 1) \cap [2^{\eta-1}, 2^\eta)$$

as the secret key sk and samples $x_i \leftarrow \mathcal{D}_{\gamma,\rho}(p)$ for $0 \leq i \leq \tau$. It then reorders the x_i so that x_0 is the largest. KeyGen(λ) restarts until x_0 is odd and $[x_0]_p$ is even. It then sets the public key $pk = \langle x_0, x_1, \cdots x_\tau \rangle$.

Encrypt(pk, m). To encrypt a message $m \in \{0, 1\}$, the encryption algorithm Encrypt(pk, m) chooses a random subset $S \subset \{1, 2, \cdots, \tau\}$ and a random integer $r \in (-2^\rho, 2^\rho)$ and outputs

$$c = [m + 2r + 2\sum_{i \in S} x_i]_{x_0}.$$

Decrypt(sk, c). The decryption algorithm Decrypt(sk, c) outputs $\tilde{m} \leftarrow (c \pmod p) \pmod 2$.

Evaluate$(pk, C, c_1, c_2, \cdots, c_t)$. Given t ciphertexts $c_1, c_2, \cdots, c_t$ as input and the boolean circuit C with t inputs, apply the addition and multiplication gates of C to the ciphertexts, perform all additions and multiplications over the integers and return the resulting integer.

Given the public key information can we recover p? Formally the approximate common divisor problem is stated as follows.

Definition 5.2.1. *(Approximate Common Divisor problem) Let $\mathcal{D}_{\gamma,\rho}(p)$ be as in equation (5.4) for security parameters (γ, η, ρ) and a randomly chosen fixed prime p of bit size η. Given polynomially many samples x_i sampled according to $\mathcal{D}_{\gamma,\rho}(p)$, the approximate common divisor problem is to recover p.*

In the x_i construction if all $r_i \neq 0$, we have a General Approximate Common Divisor (GACD) problem. If however one of the x_i's say x_0 is given as an exact multiple ($a_0 = pq_0$), then it is called Partial Approximate Common Divisor (PACD) problem. Clearly PACD problem cannot be harder than GACD problem. It is a special instance of GACD problem.

5.2.1 Exhaustive search

The ACD problem can be solved using the exhaustive search if the error terms r_i are small compared to the size of p. Given two GACD samples $x_0 = pq_0 + r_0$ and $x_1 = pq_1 + r_1$, one can recover p by computing the Greatest Common Divisor (GCD) of $x_0 - \tilde{r}_0$ and $x_1 - \tilde{r}_1$ for all possible error terms $\tilde{r}_0$ and $\tilde{r}_1$ provided that GCD($x_0 - \tilde{r}_0, x_1 - \tilde{r}_1$) is sufficiently large prime. If $r_0 = 0$, we only need to check if GCD($x_0, x_1 - \tilde{r}_1$) is sufficiently large prime for all possible values of $\tilde{r}_1$.

Let the size of r_i be given by ρ bit. Then clearly an exhaustive search takes $\tilde{O}(2^{\rho})$ and $\tilde{O}(2^{2\rho})$ GCD computations on x_i samples of size γ to solve the PACD and GACD problems respectively.

Chen et al. [CN12] improve the complexity of the exhaustive search on PACD and GACD samples to $\tilde{O}(2^{\frac{\rho}{2}})$ and $\tilde{O}(2^{\frac{3\rho}{2}})$ respectively. Consider we have PACD samples with $x_0 = pq_0$. The idea is to observe that

$$p = \text{ GCD }\left(x_0, \prod_{i=0}^{2^{\rho}-1}(x_1 \pm i) \pmod{x_0}\right). \tag{5.5}$$

Based on the observation, Chen et al. [CN12] construct a multi-evaluation polynomial $f_j(x) = \prod_{i=0}^{j-1}(x_1 \pm (x+i)) \pmod{x_0}$ of degree j with coefficients modulo x_0. Then for $\tilde{\rho} = \lfloor \frac{\rho}{2} \rfloor$

$$\prod_{i=0}^{2^{\rho}-1}(x_1 \pm i) \equiv \prod_{k=0}^{2^{\rho-\tilde{\rho}}-1} f_{2^{\tilde{\rho}}}(2^{\tilde{\rho}}k) \pmod{x_0}$$

and hence we have

$$p = \text{ GCD }\left(x_0, \prod_{k=0}^{2^{\rho-\tilde{\rho}}-1} f_{2^{\tilde{\rho}}}(2^{\tilde{\rho}}k) \pmod{x_0}\right).$$

The overall cost of computing the GCD includes $2(2^{\rho-\tilde{\rho}} - 1)$ modular multiplications and multi-evaluation of a polynomial of degree $2^{\tilde{\rho}}$ at $2(2^{\rho-\tilde{\rho}})$ points. So, Chen et al. [CN12] claim the complexity to be $\tilde{O}(\sqrt{2^{\rho}})$, which is square root of GCD exhaustive search on PACD samples. But, it incurs a memory cost of $\tilde{O}(\sqrt{2^{\rho}})$.

Applying similar method on GACD samples, the GACD problem has complexity of $\tilde{O}(2^{\frac{3\rho}{2}})$ arithmetic operations. In [CNT12], this complexity is improved to $\tilde{O}(2^{\rho})$.

GCD exhaustive search and the multi-evaluation based GCD exhaustive can be made infeasible by selecting an appropriate size bit of the errors r_i.

5.2.2 Simultaneous Diophantine approximation

(SDA-Approach) We define the simultaneous Diophantine approximation as follows.

Definition 5.2.2. *(Simultaneous Diophantine approximation) Let $a, b \in \mathbb{Z}$ and $y \in \mathbb{R}$. Let $1 < i \leq t$ for some positive integer t. The Diophantine approximation is to approximate y by a rational number a/b such that $|y - a/b| < \frac{1}{2b^2}$. The simultaneous Diophantine approximation is given many $y_i \in \mathbb{R}$ to approximate them all by the rational numbers a_i/b with $a_i \in \mathbb{Z}$ such that $|y_i - a_i/b| < \frac{1}{b^{(1+1/t)}}$.*

Van Dijk et al. [DGHV10] showed that the approximate common divisor problem can be solved using the simultaneous Diophantine approximation method. Assume we have t ACD samples $x_i = pq_i + r_i$ such that $x_0 > x_i$ for all $i > 0$ (and hence $q_0 \geq q_i$) for all values of i in that range. If one is able to find q_i then clearly a solution to the ACD problem is easy. We compute $x_i \pmod{q_i}$ to get r_i (as r_i is small). So we recover p as $p = (x_i - r_i)/q_i$. Let the rational numbers $y_i = x_i/x_0$ for $1 \leq i \leq t$. The idea with the simultaneous Diophantine approximation is to find a common approximate denominator q_0 of y_i.

Definition 5.2.3. *Let $y_i = x_i/x_0 \mid x_i, x_0 \in \mathbb{Z}$ for $1 \leq i \leq t$ be rational numbers. Let $q_0 \in \mathbb{Z}$ be the approximate common denominator of y_i. Then q_0 is of quality (ϵ, δ) if $q_0 > 0$ and the following conditions hold.*

- $q_0 \leq \epsilon x_0$ *and*
- $q_0 y_i$ *is within* $\delta/2$ *of an integer for all* $(1 \leq i \leq t)$. *In other words, there exists* u_i *such that* $|x_i/x_0 - u_i/q_0| \leq \delta/(2q_0)$.

Lemma 5.2.4. *Let* $x_i = pq_i + r_i$ *be ACD samples such that that* $x_0 > x_i$. *Let* $y_i = x_i/x_0$ *for all* $i > 0$ *and* $q_0 \in \mathbb{Z}$ *be the approximate common denominator of* y_i. *Then* q_0 *is quality* (ϵ, δ) *with* $\epsilon \leq 2^{-\eta+1}$ *and* $\delta \leq 2^{\rho-\eta+3}$.

Proof. Let $\epsilon = q_0/x_0$ then

$$\epsilon = q_0/(q_0 p + r_0) = q_0/(q_0(p + r_0/q_0)) \leq 1/(p-1) \leq 2^{-\eta+1}.$$

We also note that

$$q_0 x_i/x_0 = q_0(q_i p + r_i)/(q_0 p + r_0) = \frac{q_i(p + r_0/q_0) - (q_i/q_0)r_0 + r_i}{p + r_0/q_0} = q_i + \frac{r_i - (q_i/q_0)r_0}{p + r_0/q_0}.$$

Noting that the fractional part $\frac{r_i-(q_i/q_0)r_0}{p+r_0/q_0}$ is less than 1, the distance between $q_0(x_i/x_0)$ and the nearest integer (that is q_i) is $\delta/2 = |\frac{r_i-(q_i/q_0)r_0}{p+r_0/q_0}| \leq \frac{|r_1|+|r_0|}{p-1} \leq \frac{2^{\rho+1}}{2^{\eta-1}} = 2^{\rho-\eta+2}$ and we have $\delta \leq 2^{\rho-\eta+3}$. ☐

Taking the parameter $\frac{\delta}{2\epsilon} = \frac{2^{\rho-\eta+3}}{2.2^{-\eta+1}} = 2^{\rho+1}$, we build a lattice L with dimension $t+1$ generated by the rows of the basis matrix

$$\mathbf{B} = \begin{pmatrix} 2^{\rho+1} & x_1 & x_2 & \cdots & x_t \\ & -x_0 & & & \\ & & -x_0 & & \\ & & & \ddots & \\ & & & & -x_0 \end{pmatrix}. \tag{5.6}$$

Since the basis matrix $\mathbf{B}$ of the lattice L is given in upper triangular form, the determinant of L is easily computed as $\det(L) = 2^{\rho+1}x_0^t$.

Lemma 5.2.5. *Let* $x_i = pq_i + r_i$ *for* $1 \leq i \leq t$ *be ACD samples with* x_0 *being the largest sample. Let* $y_i = x_i/x_0$ *and* L *be the lattice with basis matrix* $\mathbf{B}$. *Then there is a lattice vector* $\mathbf{v}$ *containing a factor of the approximate common denominator* q_0 *of* y_i *in its first entry having norm approximately* $2^{\gamma-\eta+\rho+1}\sqrt{t+1}$.

Proof. Given $x_i = pq_i + r_i$ for $1 \leq i \leq t$, consider the integers values $(q_0, q_1, \cdots, q_t)$. The vector $\mathbf{v}$

$$\begin{aligned} \mathbf{v} &= (q_0, q_1, \cdots, q_t)\mathbf{B} \\ &= (2^{\rho+1}q_0, q_0x_1 - q_1x_0, \cdots, q_0x_t - q_tx_0) \\ &= (q_0 2^{\rho+1}, q_0r_1 - q_1r_0, \cdots, q_0r_t - q_tr_0) \end{aligned}$$

is in the lattice L. Since the length of each q_i is $2^{\gamma-\eta}$, the length of the first entry of the vector $\mathbf{v}$ is approximately $2^{\gamma-\eta+\rho+1}$. The length of the rest of the entries of $\mathbf{v}$, which are of the form $q_0r_i - q_ir_0$ for $1 \leq i \leq t$, is estimated to be $|q_0r_i - q_ir_0| \leq 2|q_0r_i| \approx 2^{\gamma-\eta+\rho+1}$. Taking the norm of $\mathbf{v}$ gives the result. ☐

Definition 5.2.6. *Let* $\mathbf{v}$ *be a lattice vector having norm as in Lemma 5.2.5. Then we call* $\mathbf{v}$ *a target (lattice) vector.*

To be able to break the ACD problem, we want the target vector $\mathbf{v}$ to be the shortest lattice vector. It is not possible to give a rigorous result as we have no lower bound on $\lambda_1(L)$. Instead our analysis relies on the Gaussian heuristic estimate of $\lambda_1(L)$.

Assumption 5.2.7. *Let L be the lattice generated by the rows of the basis matrix $\mathbf{B}$. Let $\mathbf{v}$ be a lattice vector having norm as in Lemma 5.2.5. Suppose*

$$||\mathbf{v}|| < \det(L)^{1/(t+1)}\sqrt{\frac{t+1}{2\pi e}}.$$

Then

1. $\lambda_1(L) = ||\mathbf{v}||$.
2. $\lambda_2(L) = \det(L)^{1/(t+1)}(1+o(1))\sqrt{\frac{t+1}{2\pi e}} = 2^{\frac{t\gamma+\rho+1}{t+1}}(1+o(1))\sqrt{\frac{t+1}{2\pi e}}$.

Lemma 5.2.8. *Let $\mathbf{v}$ be a target lattice vector. Assume Assumption 5.2.7 holds. Then $\mathbf{v}$ is the shortest lattice vector if the dimension of the lattice satisfies $\dim(L) > \frac{\gamma}{\eta}$.*

Proof. Let $\mathbf{v}$ the shortest lattice vector. By assumption (5.2.7), $||\mathbf{v}|| = \lambda_1(L) \approx 2^{\gamma-\eta+\rho+1}\sqrt{t+1}$ and the second minima satisfies

$$\lambda_2(L) = 2^{\gamma+\frac{\rho-\gamma+1}{t+1}}(1+o(1))\sqrt{\frac{t+1}{2\pi e}}.$$

Then we have $\lambda_1(L) < \lambda_2(L)$. So $2^{\gamma-\eta+\rho+1}\sqrt{t+1} < 2^{\frac{t\gamma+\rho+1}{t+1}}(1+o(1))\sqrt{\frac{t+1}{2\pi e}}$ which is implied by

$$2^{\gamma-\eta+\rho+1} < 2^{\gamma+\frac{\rho-\gamma+1}{t+1}}.$$

This is equivalent to $t+1 > \frac{\gamma-\rho-1}{\eta-\rho-1} \approx \frac{\gamma}{\eta}$. □

So t should be greater than $\frac{\gamma}{\eta}$ to ensure that the target vector $\mathbf{v}$ will likely be the shortest vector in the lattice L. However if $t < \frac{\gamma}{\eta}$, the target vector $\mathbf{v}$ will not most likely be the shortest one and it is difficult to compute using the LLL algorithm. This is because there will be many vectors of that length in magnitude in the lattice.

If t is not too large, then we can compute $\mathbf{v}$ using the LLL algorithm [LLL82]. The theoretical running time is given by $\tilde{O}(t^6 \log^3(\max_j ||b_j||))$, where $||b_j|| \approx 2^\gamma$ is the maximum norm taken over the row vectors of the basis matrix $\mathbf{B}$. Practical running times for different values of γ are given in Section 5.3.

Lemma 5.2.9. *Let $\gamma = \eta^2$ and $\dim(L)$ be given by $t = 2(\gamma/\eta)$. Assume Assumption 5.2.7 holds. Let the approximation factor of the* LLL algorithm be $\alpha = 2^{t/8}$. Then the LLL algorithm computes the target vector $\mathbf{v}$ as its shortest vector output.

Proof. Let $\dim(L)$ be given by t as in the lemma. By Lemma 5.2.5, the norm of the target vector is $||v|| = 2^{\gamma-\eta+\rho+1}\sqrt{t}$. Its approximation by α is given by

$$\alpha||\mathbf{v}|| = 2^{\frac{\eta}{4}}2^{\gamma-\eta+\rho}\sqrt{2\eta} = 2^{\gamma-\frac{3\eta}{4}+\rho}\sqrt{2\eta}.$$

We need to show $\alpha||\mathbf{v}|| < \lambda_2(L)$. By Assumption 5.2.7, we have

$$\lambda_2(L) = \det(L)^{1/(\dim(L))}(1+o(1))\sqrt{\frac{\dim(L)}{2\pi e}} \approx 2^{\gamma-\frac{\eta}{2}}\sqrt{2\eta}(1+o(1))\sqrt{\frac{\eta}{\pi e}}.$$

Up to some constant, clearly we have $\alpha||\mathbf{v}|| < \lambda_2(L)$. □

From Lemma 5.2.9, we observe that short vector outputs of the LLL algorithm depend on the size of γ and η. If $\dim(L)$ is approximately 2η for the parameter γ set to η^2, then the LLL algorithm will most likely output $\mathbf{v}$ as the shortest vector.

On the other hand if we take $\gamma = \eta^2\Omega(\lambda)$, say $\gamma = \eta^3$, and $t = 2\frac{\gamma}{\eta} = 2\eta^2$. Then by the Gaussian heuristic, short vectors are of size approximately $2^{\gamma + \frac{\rho-\gamma}{2\eta^2}}\sqrt{\frac{t}{2\pi e}} \approx 2^{\gamma - \frac{\eta}{2}}\sqrt{\frac{t}{2\pi e}}$. But the LLL algorithm with approximation factor $2^{2\epsilon\eta^2}$, where $\epsilon \in (0,1)$, can find approximation of the target vector up to size

$$2^{2\epsilon\eta^2}2^{\gamma-\eta+\rho}\sqrt{2\eta^2} = 2^{\gamma+2\epsilon\eta^2-\eta+\rho}\sqrt{2\eta^2},$$

which is much greater than $2^{\gamma-\frac{\eta}{2}}\sqrt{\frac{\eta}{\pi e}}$. As a result, it is difficult for the LLL algorithm to recover the target vector.

Once we find the target vector $\mathbf{v}$ using the LLL algorithm, we divide the first entry of $\mathbf{v}$, which corresponds to $q_0 2^{\rho+1}$, by $2^{\rho+1}$ to get q_0. Finally using q_0, we can recover p from the given ACD sample $x_0 = pq_0 + r_0$.

5.2.3 Orthogonal vectors to common divisors (NS-Approach)

Van Dijk et al. [DGHV10] discuss Nguyen and Stern's orthogonal lattice to solve the approximate common divisor problem. Assume we have ACD samples $x_i = pq_i + r_i$ for for $1 \leq i \leq (t+1)$. The idea of Nguyen and Stern's orthogonal lattice attack is to construct an orthogonal lattice $L_x^{\perp}$ such that a vector $\mathbf{z} = (z_1, \cdots z_{t+1}) \in L_x^{\perp}$ if and only if

$$z_1x_1 + z_2x_2 + \cdots + z_tx_t + z_{t+1}x_{t+1} = 0.$$

Let $L_{q,r}^{\perp}$ be another orthogonal lattice such that a vector $\mathbf{u} = (u_1, \cdots, u_{t+1}) \in L_{q,r}^{\perp}$ if and only if $u_1q_1 + \cdots u_{t+1}q_{t+1} = 0$ and $u_1r_1 + \cdots u_{t+1}r_{t+1} = 0$. Then we observe that the vector $\mathbf{u} \in L_x^{\perp}$. The final task is to find $t-1$ linearly independent vectors in $L_{q,r}^{\perp}$ that are shorter than any vector in $L_x^{\perp}$ to recover $q = (q_1, \cdots, q_{t+1})$ and $r = (r_1, \cdots, r_{t+1})$ and hence p.

Nguyen and Stern's orthogonal lattice attack to the ACD problem requires finding a vector which is both orthogonal to q and r. In a similar way to Nguyen and Stern's orthogonal lattice attack, in [DT14] it is discussed that if we can find orthogonal vectors to q only, then we can recover p as a solution to the ACD problem. As the requirement for a vector which is also orthogonal to r is eased, we focus on the latter method and we denote it as the NS-Approach. The NS notation is to denote Nguyen and Stern's first idea on orthogonal lattices to solve the ACD problem.

Suppose $(x_i = pq_i + r_i)$ for $1 \leq i \leq t$. If we can find short orthogonal vectors to $\mathbf{q} = (q_1, q_2, \cdots, q_t)$ [DT14], then we can directly find the error terms r_i to ultimately recover p as a solution to the approximate divisor problem. The method proceeds by building a t-dimensional lattice L. Then running the LLL algorithm on the lattice L to find enough short vectors for basis of L to form a system of $t-1$ linear equations in t variables of the r_i.

In the traditional approach, we consider a lattice spanned by the kernel of the matrix obtained from the system of linear equations to finally build a lattice of embedding to recover r_i. Instead we give a simplified algorithm which has the added advantage of being easier to analyse in practice.

Assume $x_t \geq x_{t-1} \geq \cdots \geq x_1$. Let the lattice L have basis given by the rows of the matrix

$$\mathbf{B} = \begin{pmatrix} 1 & & & & & x_1 \\ & 1 & & & & x_2 \\ & & 1 & & & x_3 \\ & & & \ddots & & \vdots \\ & & & & 1 & x_{t-1} \\ & & & & & x_t \end{pmatrix}. \tag{5.7}$$

Any vector $\mathbf{v}$ in the lattice L is of the form $(u_1, u_2, \cdots, u_t)\mathbf{B} = (u_1, u_2, \cdots, u_{t-1}, \sum_{i=1}^{t} u_i x_i)$ for integer values u_i. The last entry of the vector $\mathbf{v}$ gives the relation $\sum_{i=1}^{t} u_i x_i = \sum_{i=1}^{t} u_i r_i \pmod{p}$ (since $x_i = pq_i + r_i$). To exploit the relation, we need to construct a system of linear equations involving the r_i over the integers to recover r_i or q_i to finally recover p. So we require the relation to hold over integers,

$$\sum_{i=1}^{t} u_i x_i = \sum_{i=1}^{t} u_i r_i, \tag{5.8}$$

without the mod operation.

Theorem 5.2.10. *Let $x_i = pq_i + r_i$ for $1 \leq i \leq t$ be ACD samples such that GCD$(q_1, q_2, \cdots, q_t) = 1$. Let $\mathbf{B}$ be the basis matrix of L as given above. Suppose there exists a basis matrix $\tilde{\mathbf{B}}$ for the lattice L with all basis vectors short enough, with norm less than $2^{\eta-\rho-2-\log_2 t}$. Let the i^{th} row of the matrix $\tilde{\mathbf{B}}$ be given by $u_i\mathbf{B}$, where the integer entries $u_{i,j}$ form the rows of a matrix $\mathbf{U}$. Consider the linear system of equations obtained from the last entry of each row of the matrix $\tilde{\mathbf{B}}$.*

$$\begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,t} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,t} \\ u_{3,1} & u_{3,2} & \cdots & u_{3,t} \\ \vdots & \vdots & \cdots & \vdots \\ u_{t-1,1} & u_{t-2,2} & \cdots & u_{t-1,t} \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_t \end{pmatrix} = \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,t} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,t} \\ u_{3,1} & u_{3,2} & \cdots & u_{3,t} \\ \vdots & \vdots & \cdots & \vdots \\ u_{t-1,1} & u_{t-2,2} & \cdots & u_{t-1,t} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_t \end{pmatrix}. \tag{5.9}$$

Then $\langle q_1, q_2, \cdots, q_t \rangle$ is a generator for the $\mathbb{Z}$-module kernel of $\mathbf{U}$.

To prove Theorem 5.2.10, we show the existence of short vectors for the basis of the lattice L.

Lemma 5.2.11. *Let $\mathbf{v} = (v_1, v_2, \cdots, v_t) = u\mathbf{B}$, where $u = (u_1, u_2, \cdots, u_t)$ for integer values u_i. Assume $||\mathbf{v}|| < 2^{\eta-\rho-3-\log_2 t}$, then*

$$\sum_{i=1}^{t} u_i q_i = 0 \quad \textit{and} \quad \sum_{i=1}^{t} u_i x_i = \sum_{i=1}^{t} u_i r_i.$$

Proof. Let $\mathbf{v} = u\mathbf{B}$ with $u = (u_1, u_2, \cdots, u_t)$. Assume $||\mathbf{v}|| < N$ for some N, then $|u_i| < N$ for $1 \leq i \leq t-1$ and $|v_t| < N$. We need to bound u_t. Observe that $u_t = (v_t - \sum_{i=1}^{t} u_i x_i)/x_t$. So

$$\begin{aligned} |u_t| &= \left| (v_t - \sum_{i=1}^{t} u_i x_i)/x_t \right| \\ &\leq \frac{|v_t| + (t-1)|u_i||x_i|}{|x_t|} \\ &\leq \frac{N + (t-1)N|x_i|}{|x_t|} \quad (x_t > x_i) \\ &\leq tN. \end{aligned}$$

Consider a bound on $|\sum_{i=1}^{t} u_i r_i|$.

$$\begin{aligned} |\sum_{i=1}^{t} u_i r_i| &\leq |\sum_{i=1}^{t-1} u_i r_i| + |u_t r_t| \\ &\leq (t-1)N2^{\rho} + tN2^{\rho} \\ &\leq 2tN2^{\rho}. \end{aligned}$$

By taking $N = 2^{\eta-\rho-3-\log_2 t}$, we have $2tN2^{\rho} < 2^{\eta-2}$ which implies $|\sum_{i=1}^{t} u_i r_i| < p/2$.

To prove $\sum_{i=1}^{t} u_i q_i = 0$. Assume it is not so that $p|\sum_{i=1}^{t} u_i q_i| > 2^{\eta}$. We have $v_t = \sum_{i=1}^{t} u_i x_i = p(\sum_{i=1}^{t} u_i q_i) + \sum_{i=1}^{t} u_i r_i$. Then $|v_t| = |\sum_{i=1}^{t} u_i x_i| < N \implies p|(\sum_{i=1}^{t} u_i q_i)| < N + |\sum_{i=1}^{t} u_i r_i|$. So

$p|(\sum_{i=1}^{t} u_i q_i)| < N + p/2$. Since $N < 2^{\eta-\rho-2-\log_2 t}$, we have $N + p/2 < p$. This is contradiction to our assumption. So we must have $\sum_{i=1}^{t} u_i q_i = 0$ which completes the proof. □

Definition 5.2.12. *Let* $\mathbf{v}$ *be a lattice vector. If* $||\mathbf{v}|| < 2^{\eta-\rho-3-\log_2 t}$*, then* $\mathbf{v}$ *is called a target vector.*

From the target vector Definition 5.2.12 and Lemma 5.2.11, we conclude that a target vector is an orthogonal vector to $\mathbf{q}$. Each target vector is a relation. We need $t-1$ target vectors to be able to solve for $r_1, \cdots, r_t$ or $q_1, \cdots, q_t$.

Assumption 5.2.13.

$$\lambda_{t-1}(L) = \det(L)^{1/(t)}\sqrt{\frac{t}{2\pi e}}.$$

Remark 5.2.14. *This is a strong assumption hoping to get* $t-1$ *short vector basis of the lattice* L *with the maximum norm less than* $2^{\eta-\rho-3-\log_2 t}$*. The LLL algorithm outputs such vectors with some success probability (see Section 5.3).*

Lemma 5.2.15. *Let* $\mathbf{v}_{t-1}$ *be a target vector with maximum norm. Then* $\mathbf{v}_{t-1}$ *is a short vector if the dimension of the lattice* $\dim(L) > \frac{\gamma}{\eta-\rho}$.

Proof. Assume Assumption 5.2.13 holds, then we require $\det(L)^{1/(t)}\sqrt{\frac{t}{2\pi e}} < 2^{\eta-\rho-3-\log_2 t}$. Ignoring $\sqrt{\frac{t}{2\pi e}}$ this is implied by

$$2^{\gamma/t} < 2^{\eta-\rho-3-\log_2 t}$$

which is equivalent to $t > \frac{\gamma}{\eta-\rho-3-\log_2 t} \approx \frac{\gamma}{\eta-\rho}$. □

So if $t = \dim(L) > \frac{\gamma}{\eta-\rho}$, the target vector $\mathbf{v}_{t-1}$ will be short.

Lemma 5.2.16. *Let* $\gamma = \eta^2$, $\rho < \eta/4$, *and* $\dim(L)$ *be given by* $t = 2(\gamma/\eta)$. *Assume Assumption 5.2.13 holds. Then the* LLL algorithm with approximation factor $\alpha = 2^{\dim(L)/8} = 2^{\frac{\eta}{4}}$ computes a target vector as its shortest vector output.

Proof. By Assumption 5.2.13, we have

$$\lambda_{t-1}(L) = \det(L)^{1/t}\sqrt{\frac{t}{2\pi e}} < 2^{\eta-\rho-3-\log_2 t}.$$

Let $||\mathbf{v}_{t-1}||$ be a target vector with maximum norm. The LLL algorithm with approximation factor α can compute a target vector of size $\alpha\lambda_{t-1}(L)$. We need to show $\alpha\lambda_{t-1}(L) < 2^{\eta-\rho-3-\log_2 t}$. By substitution

$$\alpha\lambda_{t-1}(L) = 2^{\frac{\eta}{4}}2^{\frac{\eta}{2}}\sqrt{\frac{\eta}{\pi e}} = 2^{\frac{3\eta}{4}}\sqrt{\frac{\eta}{\pi e}},$$

which is less than $2^{\eta-\rho-3-\log_2 t}$ up to some constant. □

If however we set $\gamma = \eta^3$ and $t = 2\eta^2$, then the LLL algorithm with approximation factor $2^{2\epsilon\eta^2}$ outputs an approximation of the short vector of size $2^{2\epsilon\eta^2+\frac{\eta}{2}}\sqrt{\frac{\eta}{\pi e}}$, which is greater than $2^{\eta-\rho-3-\log_2 t}$. So the LLL algorithm fails to output a target vector.

Proof. (Theorem 5.2.10) Let $\mathbf{Z}$ be the null space of the linear system of equations (5.9). Then $\mathbf{Z}$ is of dimension one. Let $\mathbf{z} = (z_1, z_2, \cdots, z_t) \in \mathbb{Z}^t$ be a basis vector for $\mathbf{Z}$ chosen such that $\mathrm{GCD}(z_1, z_2, \cdots, z_t) = 1$, then $\mathbf{z}$ satisfies the relation

$$u_{i,1}z_1 + u_{i,2}z_2 + \cdots + u_{i,t}z_t = 0.$$

But each row u_i of the matrix $\mathbf{U}$ gives the relation

$$u_{i,1}q_1 + u_{i,2}q_2 + \cdots + u_{i,t}q_t = 0.$$

This implies $(q_1, q_2, \cdots, q_t) = k(z_1, z_2, \cdots, z_t)$ for some $k \in \mathbb{Q}$. Since $q_i \in \mathbb{Z}$ and $\mathrm{GCD}(q_1, q_2, \cdots, q_t) = 1$, we must have $k = \pm 1$. So $\langle q_1, q_2, \cdots, q_t\rangle$ is a generator for the $\mathbb{Z}$-module kernel of $\mathbf{U}$. □

As a result of Theorem 5.2.10, we obtain a solution to the ACD problem. The values of $(q_1, q_2, \cdots, q_t)$ are obtained directly from the basis of the null space of the system equations (5.9). Recovering p is then immediate.

5.2.4 Orthogonal vectors to error terms (NS*-Approach)

Let $x_i = pq_i + r_i$ be ACD samples, where $1 \le i \le t$, with $x_0 = pq_0$. In Section 5.2.3, we considered short orthogonal vectors to $\mathbf{q} = (q_1, q_2, \cdots, q_t)$ to solve the ACD problem. A similar method by Van Dijk et al. is proposed in Appendix B.1 of [DGHV10]. Instead of orthogonal vectors to $\mathbf{q}$, orthogonal vectors to the vector $(1, \frac{-r_1}{R_1}, \frac{-r_2}{R_2}, \cdots, \frac{-r_t}{R_t})$, where R_i is an upper bound on each error term r_i, are used. This is also mentioned in [CS15]. We call this method the NS*-Approach.

The way the NS*-Approach works is similar to our discussion in Section 5.2.3. A system of linear equations will be built involving the variables r_i from short vector outputs of the LLL algorithm applied on a lattice L. If the short vectors used to construct the system of equations are sufficiently small enough, the kernel reading of the corresponding system of equations reveals the values of $\mathbf{q}$.

Assume the same bound R for all error bounds R_i. Let L be a lattice with basis matrix

$$\mathbf{B} = \begin{pmatrix} x_0 & & & & & \\ x_1 & R & & & & \\ x_2 & & R & & & \\ x_3 & & & R & & \\ \vdots & & & & \ddots & \\ x_t & & & & & R \end{pmatrix}. \tag{5.10}$$

Notice that each row of $\mathbf{B}$ corresponds $x_i - r_i \equiv 0 \pmod p$. As $\mathbf{B}$ is a triangular matrix, $\det(L) = x_0 R^t$.

The lattice L is similar to the lattice considered in Section 5.2.3 with the diagonals scaled by the error bound R. In Appendix B.1 of [DGHV10], the first row of the matrix $\mathbf{B}$, $x_0 = pq_0$ with error term 0 is not considered. For the sake of bounding the determinant easily, we include it in our analysis.

Any vector $\mathbf{v} = (v_0, v_1, \cdots, v_t) \in L$ is of the form

$$\mathbf{v} = (u_0, u_1, \cdots, u_t)\mathbf{B} = (\sum_{i=0}^{t} u_i x_i, u_1 R, u_2 R, \cdots, u_t R),$$

where $u_i \in \mathbb{Z}$. The main observation of Van Dijk et al. [DGHV10] is

$$v_0 - \sum_{i=1}^{t} \frac{v_i}{R} r_i = \sum_{i=0}^{t} u_i x_i - \sum_{i=1}^{t} \frac{u_i R}{R} r_i = \sum_{i=1}^{t} u_i (x_i - r_i) = 0 \pmod p. \tag{5.11}$$

To be able to form a system of linear equations involving the variables r_i, we require equation (5.11) to hold over the integers. Clearly if $|v_0 - \sum_{i=1}^{t} \frac{v_i}{R} r_i| < p/2$,

$$v_0 - \sum_{i=1}^{t} \frac{v_i}{R} r_i = 0. \tag{5.12}$$

So we need the following lemma to satisfy this condition.

Lemma 5.2.17. *Let* $\mathbf{v} = (u_0, u_1, u_2, \cdots, u_t)\mathbf{B}$. *Let* $||\mathbf{v}|| < 2^{\eta - 2 - \log_2(t+1)}$. *Then*

$$|v_0 - \sum_{i=1}^{t} \frac{v_i}{R} r_i| < p/2 \quad \textit{and} \quad \sum_{i=1}^{t} u_i q_i = 0.$$

Proof. Let $\mathbf{v} = (v_0, v_1, \cdots, v_t) = (\sum_{i=0}^{t} u_i x_i, u_1 R, u_2 R, \cdots, u_t R)$. Let $||\mathbf{v}|| < N$ for some positive real number N. Then $|v_0| < N$ and $|u_i R| < N$ for $1 \le i \le t$. Thus

$$|v_0 - \sum_{i=1}^{t} \frac{v_i}{R} r_i| < |v_0| + |\sum_{i=1}^{t} u_i r_i| < N + tN.$$

Taking $N = 2^{\eta - 2 - \log_2(t+1)}$, we have $(t+1)N < 2^{\eta-2}$. So $|v_0 - \sum_{i=1}^{t} \frac{v_i}{R} r_i| < p/2$.

To prove $\sum_{i=0}^{t} u_i q_i = 0$, suppose $\sum_{i=0}^{t} u_i q_i \neq 0$ so that $p|\sum_{i=0}^{t} u_i q_i| > 2^{\eta}$. Since $x_i = pq_i + r_i$, we have

$$p|\sum_{i=0}^{t} u_i q_i| < |\sum_{i=0}^{t} u_i x_i| + |\sum_{i=1}^{t} u_i r_i| < 2^{\eta - 2 - \log_2(t+1)} + (t+1)2^{\eta - 2 - \log_2(t+1)} < p.$$

This is contradiction to our assumption. We must have $\sum_{i=0}^{t} u_i q_i = 0$. $\square$

Definition 5.2.18. *Let* $\mathbf{v}$ *be a lattice vector in* L. *If* $||\mathbf{v}|| < 2^{\eta-2-\log_2(t+1)}$, *then* $\mathbf{v}$ *is called a target vector.*

As can be observed from equation (5.12) and Lemma 5.2.17, if $\mathbf{v} = (u_0, u_1, \cdots, u_t)\mathbf{B}$ is a target vector having norm less than $2^{\eta-2-\log_2(t+1)}$, then $\mathbf{v}$ is an orthogonal vector to

$$(1, \frac{-r_1}{R_1}, \frac{-r_2}{R_2}, \cdots, \frac{-r_t}{R_t}),$$

and implicitly the vector $(u_0, u_1, \cdots, u_t)$ is orthogonal to $(q_0, q_1, q_2, \cdots, q_t)$. We need $t-1$ such target vectors to form a linear system of equations of the form $\sum_{i=1}^{t} u_i x_i = \sum_{i=1}^{t} u_i r_i$ in the unknown variables r_i.

Theorem 5.2.19. *Let* $x_i = pq_i + r_i$ *be ACD samples, where* $1 \le i \le t$, *such that GCD*$(q_1, q_2, \cdots, q_t) = 1$. *Let* $\mathbf{B}$ *be the basis matrix of* L *as given above. Suppose there exists a basis matrix* $\tilde{\mathbf{B}}$ *for the lattice* L *with all basis vectors short enough, with norm less than* $2^{\eta-1-\log_2(t+1)}$. *Let the* i^{th} *row of the matrix* $\tilde{\mathbf{B}}$ *be given by* $u_i\mathbf{B}$, *where* $u_{i,j} \in \mathbb{Z}$ *form the rows of a matrix* $\mathbf{U}$. *Consider the linear system of equations*

$$\begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,t} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,t} \\ u_{3,1} & u_{3,2} & \cdots & u_{3,t} \\ \vdots & \vdots & \cdots & \vdots \\ u_{t-1,1} & u_{t-2,2} & \cdots & u_{t-1,t} \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_t \end{pmatrix} = \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,t} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,t} \\ u_{3,1} & u_{3,2} & \cdots & u_{3,t} \\ \vdots & \vdots & \cdots & \vdots \\ u_{t-1,1} & u_{t-2,2} & \cdots & u_{t-1,t} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_t \end{pmatrix}. \quad (5.13)$$

Then $\langle q_1, q_2, \cdots, q_t \rangle$ *is a generator for the* $\mathbb{Z}$*-module kernel of* $\mathbf{U}$.

Proof. We follow similar steps used to prove Theorem 5.2.10. Note that given a lattice vector $\mathbf{v} = (v_0, v_1, \cdots, v_t) = (u_0, u_1, \cdots, u_t)\mathbf{B}$, then $u_i = \frac{v_i}{R}$ for $1 \le i \le t$. $\square$

This method relies on finding $t-1$ short vectors. As discussed in Section 5.2.3, by Assumption 5.2.13 for the target vector Definition 5.2.18, the LLL algorithm outputs such short vectors with some success probability (see Section 5.3) enough to build a linear system of equations (5.13). Then we directly recover the values of $(q_1, \cdots, q_t)$ from the kernel of the system of equations. It is then immediate to recover p a solution to the ACD problem.

5.2.5 Multivariate polynomial equations method (CH-Approach)

Howgrave-Graham [HG01] studied the PACD problem given two ACD sample inputs, $N = pq_1$ and $a = pq_2 + r_1$. The idea is based on solving modular univariate linear equations of the form $a + x = 0 \pmod{p}$ for unknown p. A solution $x = r_1$ to the modular equation is expected to be small. The key observation is based on the following lemma.

Lemma 5.2.20. *(Howgrave-Graham) Let* $Q(x_1, \cdots, x_m) \in \mathbb{Z}[x_1, \cdots, x_m]$ *be an integer polynomial given as*

$$Q(x_1, \cdots, x_m) = \sum_{j_1, \cdots, j_m} Q_{j_1 \cdots j_m} x_1^{j_1} \cdots x_m^{j_m}.$$

Define the norm of a polynomial $|Q(x_1, \cdots, x_m)|$ *to be* $\sqrt{\sum Q^2_{j_1 \cdots j_m}}$. *Assume that* Q *has* ω *monomials. If*

- $Q(r_1, \cdots, r_m) = 0 \pmod{p^k}$ *for* $|r_1| < R_1, \cdots, |r_m| < R_m$ *, where the* R_i *error bounds and*
- $|Q(R_1x_1, \cdots, R_mx_m)| < \frac{p^k}{\sqrt{\omega}}$,

then $Q(r_1, \cdots, r_m) = 0$ *over the integers.*

In [DGHV10] (see Appendix **B**.2), Howgrave-Graham's approach is generalized to a multivariate version of the problem which is an extension of Coppersmith's method [Cop96b]). A more general multivariate approach is mentioned in [CH13] and we focus on the latter. Let $\beta \in (0, 1)$ and $N = pq_0$ such that $p = N^\beta$. Let $a_i = q_ip + r_i$ for $1 \leq i \leq m$. Clearly we have $a_i - r_i \equiv 0 \pmod{p}$. The idea of [CH13] is based on the observation that the products $(a_i - r_i)^2 \equiv 0 \pmod{p^2}$, $(a_i - r_i)(a_j - r_j) \equiv 0 \pmod{p^2}$ and so on and so forth. In general polynomials constructed as a linear combination of the products modulo the power of p have the same roots.

One can then construct polynomials $Q_1, Q_2, \cdots Q_m$ in m variables such that $Q(r_1, \cdots, r_m) \equiv 0 \pmod{p^k}$ for some k. The hope is now if the constructed polynomials are sufficiently small when evaluated at small integer values, then with high probability solving the system of equations over $\mathbb{Q}$ gives candidates for the roots $r_1, \cdots, r_m$.

Assume k, ℓ, t are parameters to be optimized. If the constructed polynomials have small coefficients, the congruence modulo p^k holds over the integers. The idea is to build Q such that $|Q(r_1, \cdots, r_m)| < p^k$. To ensure that Cohn et al. [CH13] construct the polynomials

$$Q(r_1, \cdots, r_m) \equiv 0 \pmod{p^k} \tag{5.14}$$

as integer linear combinations of the products

$$(x_1 - a_1)^{i_1} \cdots (x_m - a_m)^{i_m} N^\ell$$

such that $i_1 + \cdots + i_m + \ell \geq k$ in the indeterminate variables $x_1, \cdots, x_m$.

Accordingly Cohn et al. [CH13] consider the lattice L generated by the coefficient row vectors of the products

$$(R_1x_1 - a_1)^{i_1} \cdots (R_mx_m - a_m)^{i_m} N^\ell, \tag{5.15}$$

such that $i_1 + \cdots + i_m \leq t$ and $\ell = max(k - \sum_j i_j, 0)$.

Let $f = (R_1x_1 - a_1)^{i_1} \cdots (R_mx_m - a_m)^{i_m} N^\ell$. Let $f_{[i_1,_2,\cdots,i_j]}$ denote f evaluated at $(i_1, i_2, \cdots i_m)$ (this is mentioned in [TK13]). If one orders the monomials occurring in $f_{[i_1,_2,\cdots,i_j]}$, say with degree reverse lexicographic ordering, then the basis matrix of the corresponding lattice L is lower triangular. For example, for choices of $t = 3, m = 2, k = 1$, the corresponding basis matrix $\mathbf{B}$ of the lattice L is of the form

$$\mathbf{B} = \begin{array}{c} f_{[i_1,i_2]} \\ f_{[0,0]} \\ f_{[1,0]} \\ f_{[0,1]} \\ f_{[2,0]} \\ f_{[1,1]} \\ f_{[0,2]} \\ \vdots \\ f_{[0,3]} \end{array} \begin{array}{c} \begin{array}{ccccccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & \dots & x_2^3 \end{array} \\ \begin{pmatrix} N & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -a_1 & R_1 & 0 & 0 & 0 & 0 & \dots & 0 \\ -a_2 & 0 & R_2 & 0 & 0 & 0 & \dots & 0 \\ a_1^2 & -2a_1R_1 & 0 & R_1^2 & 0 & 0 & \dots & 0 \\ a_1a_2 & -a_2R_1 & -a_1R_2 & 0 & R_1R_2 & 0 & \dots & 0 \\ a_2^2 & 0 & -2a_2R_2 & 0 & 0 & R_2^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_2^3 & 0 & 3a_2^2R_2 & 0 & 0 & -3a_2R_2^2 & \dots & R_2^3 \end{pmatrix} \end{array}. \tag{5.16}$$

Lemma 5.2.21. *Assume* $m > 1$ *and* $t > 1$*. Then the dimension of the lattice* L *generated by the coefficient row vectors of the products given by equation (5.15) is* $\binom{t+m}{m}$.

Proof. The dimension of the lattice is clearly the number of possible polynomials of the form $(R_1x_1 - a_1)^{i_1} \cdots (R_mx_m - a_m)^{i_m} N^\ell$ in the variables $(x_1, \cdots, x_m)$. So we need to count the possible number of combinations of the exponents i_j, where $i_j \geq 0$, so that $0 \leq i_1 + \cdots + i_m \leq t$. Assigning $t+1$ values $(0, 1, 2, \cdots, t)$ to m exponents i_j can be done in $\binom{m+t}{m}$ ways.

Equivalently, we count the number of non-negative integer solutions to the equation $i_1 + \cdots + i_m = \tilde{t}$ in the variables i_j. It has $\binom{m+\tilde{t}-1}{\tilde{t}}$ possible number of solutions. Adding all possible number of solutions for $0 \leq \tilde{t} \leq t$ gives the result. Note that since $\binom{m+t}{m} = \binom{m+t-1}{t} + \binom{m+t-1}{t-1}$,

$$\binom{m-1}{0} + \binom{m}{1} + \cdots + \binom{m+t-1}{t} = \binom{m+t}{m}.$$

□

Lemma 5.2.22. *Let the error bounds R_i have the same value R. Then the determinant of the lattice L generated by the coefficient row vectors of the products given by equation (5.15) is*

$$\det(L) = R^{\binom{t+m}{m}\frac{mt}{m+1}} N^{\binom{k+m}{m}\frac{k}{m+1}} = 2^{\binom{t+m}{m}\frac{\rho mt}{m+1} + \binom{k+m}{m}\frac{\gamma k}{m+1}}.$$

Proof. $\det(L) = N^{S_N} R^{mS_R}$, where S_N is the sum of exponents of N and S_R is the sum of exponents of R_i. Lemma 5.2.21 implies that there are in total $\binom{t+m}{m}$ monomials with $\binom{m+i-1}{i}$ of them having exponent i. This implies that on average each R_i has exponent $i\binom{m+i-1}{i}/m = \binom{m+i-1}{i-1}$. Summing up for $1 \leq i \leq t$ gives the total exponent of R_i. So we have

$$S_R = \binom{m}{0} + \binom{m+1}{1} + \cdots + \binom{m+t-1}{t-1} = \binom{m+t}{t-1} = \binom{m+t}{m}\frac{\binom{m+t}{t-1}}{\binom{m+t}{m}} = \binom{m+t}{m}\frac{t}{m+1}.$$

The exponent of N in each monomial expression is ℓ, where $\ell = max(k - \sum_j i_j, 0)$. In other words we demand $(i_1 + i_2 + \cdots + i_m \leq k)$. A similar analysis gives the exponent of N to be $S_N = \binom{m+k}{m}\frac{k}{m+1}$. Substituting N and R by their size estimates 2^γ and 2^ρ respectively gives the result. □

If $|Q(r_1, \cdots, r_m)| < p^k$, clearly equation 5.14 holds over the integers. We need to estimate the norm of the corresponding lattice vector. Let $Q(x_1, \cdots, x_m) = \sum_{j_1, \cdots, j_m} (Q_{j_1 \cdots j_m} x_1^{j_1} \cdots x_m^{j_m})$. Its corresponding lattice vector $\mathbf{v}$ is

$$\mathbf{v} = \sum_{j_1, \cdots, j_m} (Q_{j_1 \cdots j_m} R_1^{j_1} \cdots R_m^{j_m}).$$

We note that $|Q(r_1, \cdots, r_m)| \leq |\mathbf{v}|_1$, where $|\mathbf{v}|_1$ is the ℓ_1 norm of $\mathbf{v}$. Indeed

$$\begin{aligned} |Q(r_1, \cdots, r_m)| &\leq \sum_{j_1, \cdots, j_m} |Q_{j_1 \cdots j_m}||r_1|^{j_1} \cdots |r_m|^{j_m} \\ &\leq \sum_{j_1, \cdots, j_m} |Q_{j_1 \cdots j_m}| R_1^{j_1} \cdots R_m^{j_m} \\ &= |\mathbf{v}|_1. \end{aligned}$$

So we define a target vector as follows.

Definition 5.2.23. *Let L be the lattice generated by the coefficient row vectors of the products given by equation (5.15). Let $\mathbf{v}$ be a vector in L. If $|\mathbf{v}|_1 < p^k$ for some positive integer k then $\mathbf{v}$ is called a target vector.*

Each target vector gives a relation. So if we have m relations, we solve the corresponding polynomial equations in m variables. We need a heuristic assumption to get m such target vectors.

Assumption 5.2.24. *Let L be the lattice generated by the coefficient row vectors of the products given by equation (5.15). Then $\lambda_m(L) = \det(L)^{1/(\omega)}\sqrt{\frac{\omega}{2\pi e}}$, where $\omega = \binom{t+m}{m}$ is the dimension of L.*

Lemma 5.2.25. *Let L be the lattice generated by the coefficient row vectors of the products given by equation (5.15). Then $\lambda_m(L)$ is short if $\omega = \binom{t+m}{m} > \frac{\binom{k+m}{m}\gamma}{(m+1)(\eta-\rho)}$.*

Proof. If Assumption 5.2.24 hods, then we require $\lambda_m(L) < p^k$ which implies $\log_2 \det(L) < \omega k\eta$. So we have

$$\omega\rho\frac{mt}{m+1} + \gamma\binom{k+m}{m}\frac{k}{m+1} < k\omega\eta$$

which is implied by $\gamma\binom{k+m}{m}\frac{1}{m+1} < \omega(\eta - \rho(t/k))$. This is equivalent to

$$\omega = \binom{t+m}{m} > \frac{\binom{k+m}{m}\gamma}{(m+1)(\eta-\rho(t/k))} \approx \frac{\binom{k+m}{m}\gamma}{(m+1)(\eta-\rho)}.$$

□

By Lemma 5.2.25 for $\omega > \frac{\binom{k+m}{m}\gamma}{(m+1)(\eta-\rho)}$, the first m output vectors $\mathbf{v}_i$ of the LLL algorithm satisfy $|\mathbf{v}_i| < p^k$ giving us polynomial relations between $r_1, \cdots, r_m$. More specifically we write $\mathbf{v} = (u_1, \cdots, u_\omega)$ and consider the ω monomials $(1, x_1, x_2, x_1^2, x_1x_2, x_2^2, \cdots, x_m^t)$ in degree reverse ordering. Then the corresponding polynomial to lattice vector $\mathbf{v}$ is

$$Q(x_1, x_2, \cdots, x_m) = \sum_{i=1}^{\omega} \frac{u_i}{R_1^{j_1}\cdots R_m^{j_m}} x_1^{j_1}\cdots x_m^{j_m}.$$

We collect m such independent polynomial equations. The system of equations coming from these relations can then be solved using the F4 [Fau99] or F5 [Fau02] Gröbner basis algorithms to directly find all $r_1, \cdots, r_m \in \mathbb{Z}$. Note that the first m output of the LLL algorithm do not necessarily give an algebraic independent vectors. In this case we add some next output vectors of the LLL algorithm with ℓ_1 norm less than p^k (if there are any). Alternatively we factorize the polynomial equations to get algebraic independent polynomial equations. Finally with high probability we recover p by computing

$$\gcd(N, a_1 - r_1, a_2 - r - 2, \cdots, a_m - r_m).$$

The drawback of the CH-Approach is that we may not find enough independent polynomial equations. Our experiment (see Table 5.1) shows that indeed this is the case. As a result, the running time of the Gröbner basis part is stuck even for small parameters.

5.3 Comparison of algorithms for the ACD problem

The approximate common divisor problem is currently a hard problem for appropriate parameter settings. We have discussed that there are cryptographic applications that exploit the hardness of the ACD problem. The DGHV homomorphic [DGHV10] encryption over the integers is a particular example. For an ACD sample $x_i = pq_i + r_i$, recall that γ, η, ρ are the bit size of the parameters x_i, p, r respectively. In [DGHV10], the parameters are set as $\gamma = \eta^2\Omega(\lambda)$, $\eta = \lambda^2$ and $\rho = \lambda$ for security parameter λ.

The security proof analysis of the DGHV homomorphic [DGHV10] encryption and other variants such as [CMNT11] are based on the complexity analysis of the different algorithms to solve the ACD computational problem. These algorithms are in turn based on the worst-case performance of the LLL algorithm. It is important to analyze the current most effective algorithm to solve the ACD problem from practical point of view.

The CH-Approach (see Section 5.2.5) reduces solving the ACD problem to solving multivariate polynomial equations. For this approach to be successful, the dimension of the lattice L must satisfy $\dim(L) > \frac{\binom{k+m}{m}\gamma}{(m+1)(\eta-\rho)} \approx \gamma/(\eta-\rho)$ (for some parameters $k, m > 1$). As γ is set to be greater than η^2 (see [CS15] for tighter bounds), the CH-Approach becomes infeasible to solve the ACD problem for such parameter settings.

The SDA-Approach (see Section 5.2.2) solves the ACD problem using simultaneous Diophantine approximation method. The dimension of the lattice required is greater than γ/η. As explained if the ratio of these parameters is too large, the LLL algorithm cannot produce the required output. Similarly in the case of NS-Approach and NS*-Approach, see Sections 5.2.3 and 5.2.4 respectively, the dimension of the lattice required is greater than $\gamma/(\eta-\rho)$.

One can see that the lattice dimension requirement $\dim(L) > \gamma/\eta$ in the case of SDA-Approach and $\dim(L) > \gamma/(\eta-\rho)$ in the case of CH-Approach, NS-Approach and NS*-Approach are the limiting factor of the LLL algorithm. In all cases if the ratio of γ to η is large, the LLL algorithm fails to output target vectors in the lattice constructed for each approach.

Assuming parameters are not large as in [DGHV10], we ask ourselves which algorithm for the approximate common divisor problem is best in practice? We ran experiments for some fixed values of parameters. Basically since the limiting factor is the ratio γ/η, we fix p to be of size $\eta = 100$ and we let the size of x_i, γ grow linearly up to η^2. In other words, the size of the q_i goes from 1 to η as shown in Table 5.1.

Table 5.1: Comparison of NS-Approach, NS*-Approach, SDA-Approach and CH-Approach. The common parameters ρ and γ indicate the size of error terms r_i and the size of the GACD samples x_i in bits (respectively) for a fixed common parameter prime p of size $\eta = 100$ bits. Time indicates the running time of the four approaches, $\dim(L)$ represents the dimension of the lattice constructed required for each approach and Success denotes the percentage of times each algorithm recovers the correct p. The experiment is repeated 100 times except for those indicated with $*$ and $\star$, which in these cases are repeated only once and 10 times respectively. The notation $\times$ indicates the running time is too high for us to get any useful data.

			NS-Approach		NS*-Approach		SDA-Approach		CH-Approach with $(t, k) = (3, 2)$		
γ	ρ	$\dim(L)$	Time	Success	Time	Success	Time	Success	$\dim(L)$	Time	Success
150	10	12	0.009	100%	0.002	100%	0.008	100%	35	0.199	100%
	49	13	0.009	100%	0.002	100%	0.008	48%	35	0.134	100%
	50	13	0.009	100%	0.009	100%	0.009	0%	35	0.132	100%
	58	14	0.009	100%	0.010	100%	0.009	0%	35	0.110	100%
300	10	13	0.007	100%	0.005	100%	0.010	100%	35	0.321	100%
	58	17	0.010	100%	0.007	100%	0.013	100%	35	0.141	100%
	90	40	0.036	2%	0.056	78%	0.086	72%	35	0.355	0%
	92	48	0.052	0%	0.088	26%	0.136	18%	35	0.345	0%
600	10	17	0.022	100%	0.016	100%	0.025	100%	35	1.108	100%
	30	19	0.021	100%	0.016	100%	0.032	100%	35	0.945	100%
	85	50	0.137	1%	0.276	56%	0.591	58%	35	0.205	0%
	88	60	0.213	0%	0.488	1%	1.010	1%	35	0.179	0%
1200	10	23	0.065	100%	0.059	100%	0.115	100%	56	53.060	100%
	20	25	0.081	100%	0.072	100%	0.152	100%	56	10.401	100%
	75	58	0.589	8%	0.929	61%	2.926	68%	56	8.515	0%
	80	70	0.879	0%	1.780	1%	5.225	85%	56	6.955	0%
2400	10	37	0.513	100%	0.476	100%	1.565	100%		$\times$	$\times$
	50	58	2.444	94%	2.133	97%	8.181	100%		$\times$	$\times$
	70	90	8.907	0%	9.823	0%	36.164	2%		$\times$	$\times$
5000	10	66	8.205	100%	8.154	100%	44.372	100%		$\times$	$\times$
	20	73	12.388	100%	11.200	100%	63.273	100%		$\times$	$\times$
	40	94	28.833	9%	29.135	33%	137.928	50%		$\times$	$\times$
6000	10	77	17.169	94%	17.315	98%	97.953	100%		$\times$	$\times$
	15	81	20.640	89%	21.212	98%	118.114	90%		$\times$	$\times$
	40	110	63.245	0%	63.121	0%	403.000	0%		$\times$	$\times$
7000	10	88	33.231	100%	33.894	100%	208.151	80% $\star$		$\times$	$\times$
	15	92	41.194	78%	41.777	86%	263.172	90% $\star$		$\times$	$\times$
	30	110	78.496	1%	81.022	1%	497.381	0% $\star$		$\times$	$\times$
8000	10	99	65.523	100%	58.707	90%	464.465	100% $\star$		$\times$	$\times$
	15	104	84.627	80%	72.206	70%	718.458	50% $\star$		$\times$	$\times$
	20	120	111.400	70% $\star$	104.652	90% $\star$	2414.070	100% *		$\times$	$\times$
9000	10	120	122.516	80% $\star$	103.297	40% $\star$	4108.840	100% *		$\times$	$\times$
	15	126	236.098	50% $\star$	239.258	60% $\star$	3585.130	100% *		$\times$	$\times$
	20	133	392.627	40% $\star$	395.834	90% $\star$	5246.490	100% *		$\times$	$\times$
10^4	10	131	304.395	40%$\star$	283.870	100% $\star$	5917.750	100% *		$\times$	$\times$
	15	138	618.203	30% $\star$	306.982	100% $\star$	5472.070	100% *		$\times$	$\times$
	20	145	1136.870	20%$\star$	934.202	60%$\star$	8151.022	100% *		$\times$	$\times$

5.3.1 Experimental observation

In Table 5.1, Time refers to the total running time of each approach to solve the ACD problem and Success refers to how accurate each approach outputs the correct solution p for a given error size. The common parameters of the four approaches ρ, η and γ refer to the size of the error terms, the size of the prime p and the size of ACD samples in bits respectively and $\dim(L)$ denotes the dimension of the lattice used.

Given a fixed dimension $\dim(L)$, an algorithm for solving ACD problem is best if it has better running time with maximum accuracy for a given error size. An algorithm capable of handling larger errors r_i is also preferable.

Considering the running time of the four approaches, we clearly observe from our experiment that the NS-Approach and NS*-Approach are the fastest ones. In other words, the two orthogonal lattice based approaches have low running times compared with the other two. As the dimension of the lattice increases, the SDA-Approach has slower running time than the running time of the NS-Approach and NS*-Approach. This is mainly because the volume of the lattice constructed for the SDA-Approach is larger than the NS-Approach and NS*-Approach.

The CH-Approach has the worst running time compared with the others. Its running time is extremely slow. As it can be observed from the Table 5.1 for most parameter values, the algorithm is extremely slow!. This is mainly because the volume of the lattice constructed is higher at least by a factor of $\frac{k\binom{k+m}{m}}{m+1}$ than the other approaches. Consequently the dimension of the lattice required to output short vectors is greater than $\frac{\binom{k+m}{m}\gamma}{(m+1)(\eta-\rho)}$ which is higher than the dimension requirement for the other approaches for the same parameters considered. Hence as Table 5.1 shows, the CH-Approach is only effective when the size of the ACD samples indicated as γ is close to the size of the prime p. In other words, the q_i's are so small. But in practice the q_i's are large. So the CH-Approach is not effective algorithm to solve ACD problem. Moreover, solving multivariate polynomials using the F4 or F5 Gröbner basis algorithms is slow.

So we conclude that orthogonal lattice based approaches (the NS-Approach and NS*-Approach) followed by the SDA-Approach are best attacks available today against the ACD problem. The CH-Approach is always slower than the four approaches except when the q_i are so small.

5.4 Pre-processing of the ACD samples

The limiting factor of the different algorithms to solve the approximate common divisor problem for parameters settings as in [DGHV10] is the ratio of the size of the x_i given in γ bits to the ratio of the size of p given in η bits is large. This ultimately is the limiting factor of the LLL algorithm whose complexity is dependent on the ratio of these parameters.

To reduce the size of the ACD samples x_i, we suggest a pre-processing step of the ACD samples. Assume we have $\tau = \gamma + \rho$ ACD samples [DGHV10]. Then observe that the samples $x_{i+1} - x_i = p(q_{i+1} - q_i) + r_{i+1} - r_i$ for $1 \leq i \leq \tau$ obtained by subtracting consecutive samples of the x_i are also ACD samples potentially with size $\leq \gamma - 1$.

Let Δ_k represent the ACD samples at round k with $\Delta_k[i]$ representing an ACD sample in the i^{th} position. Define $\Delta_0[i] = x_i$. Assume the x_i are in base-10 numeration system. To get Δ_k ACD samples, we eliminate the most significant digit of the Δ_{k-1} ACD samples as follows. Select 9 values with distinct most significant digits of the Δ_{k-1} ACD samples. Assume $\Delta_{k-1}[1], \Delta_{k-1}[2], \cdots, \Delta_{k-1}[9]$ are consecutive ACD samples having distinct most significant digits.

Consider now Δ_{k-1} ACD samples for $10 \leq i \leq \#\Delta_{k-1}$. For each $\Delta_{k-1}[i]$ ACD sample, we match its most significant digit with the most significant digits of $\Delta_{k-1}[j]$ ACD samples, where $1 \leq j \leq 9$. Then taking positive difference of the samples whose most significant digits match gives $\Delta_k[i]$.

We note that the number of Δ_k ACD samples is at most 10 less than Δ_{k-1} ACD samples. We continue this operation until we are able to solve the ACD problem using the orthogonal lattice based methods. The following Algorithm 2 generalizes the idea of the pre-processing step of the ACD samples.

The number of digits of x_i is polynomial in the size of the input. So we have a polynomial time running algorithm. The disadvantage of the pre-processing step of the ACD samples is that the error size might increase. Precisely at round k, in the worst-case the error size is scaled up by a factor of 2^k.

Lemma 5.4.1. *Let k be the number of times the outer loop of Algorithm 2 runs. Assume algorithms for solving the ACD problem can handle up to $\eta/2$ error size. Then any of these algorithms making use of the pre-processing of the ACD samples step has a negligible success probability if $k > \eta - 2\rho$.*

Algorithm 2 Pre-processing of the ACD samples

```
Input: τ GACD sample x_i and a base 2^b.
Δ_0[i] ← x_i.
d ← Digits(2^γ) % Number of digits of an ACD sample.
s ← bd % Initial number of shifts to get the most significant digit.
k ← 1 % Number of rounds.
while Until we get enough reduced ACD samples do
    T[j] ← 0 % Initialize temporary storage of size b.
    while i ≤ #Δ_{k-1} do
        MSB ← ShiftRight(Δ_{k-1}[i], s) % Compute most significant digits of Δ_{k-1}[i].
        if MSB is zero then
            Add Δ_{k-1}[i] to our new reduced sample Δ_k.
        else if T[MSB] is zero then
            T[MSB] = Δ_{k-1}[i].
        else
            Take positive difference of Δ_{k-1}[i] and T[MSB] and add it to Δ_k.
    s ← s − b.
    k ← k + 1.
```

Proof. The number of errors involved in each round of the Δ_k pre-processing step is 2^k. Specifically, at round k, Δ_k is of the form $\Delta_k[i] = \sum_{i=1}^{2^k} c_i x_i$, where $c_i = \pm 1$. Let $\tilde{r}_k$ be the corresponding error sum. Then $\tilde{r}_k = \sum_{i=1}^{2^k} c_i r_i$, where $c_i = \pm 1$. Since r_i is uniformly distributed on $(2^{-\rho}, 2^{\rho})$, its expected value is 0 with variance approximately equal to $\frac{1}{3}(2^{2\rho})$. Thus $\tilde{r_k}$ is normally distributed with mean 0 and variance approximately equal to $\frac{1}{3}(2^{2\rho+k})$. So the expected sum of all the errors involved in the final reduced Δ_k ACD sample is given by the standard deviation of $\tilde{r}_k$

$$\sqrt{\frac{1}{3}(2^{2\rho+k})} \approx 2^{k/2+\rho}.$$

By assumption, existing algorithms are able to solve the ACD problem with $\eta/2$ error size. So we require $2^{k/2+\rho} \leq 2^{\eta/2}$ to have a non-negligible success of probability. This implies $k \leq \eta - 2\rho$. $\square$

Suppose $k \leq \eta - 2\rho$. Then the algorithm reduces the size of the ACD samples by k digits. As a result q is smaller. But to reduce significantly, we need $k \approx \lambda^5$ which is not less than $k \leq \eta - 2\rho \approx \lambda^2 - 2\lambda$. Thus the pre-processing idea does not have a serious impact on the ACD problem.

Bibliography

[AD94] L. Adleman, J. DeMarrais. A subexponential algorithm for discrete logarithms over all finite fields. In Advances in Cryptology-CRYPTO'93, pages 147–158, 1994.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems. In Proc. 28th ACM Symp. on Theory of Computing, pages 99–108, 1996.

[Ajt98] M. Ajtai. The shortest vector problem in L2 is NP-hard for randomized reductions. In STOC'98, pages 10–19, 1998.

[AKS01] M. Ajtai, R. Kumar, D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In Proc. STOC'01, pages 601–610, 2001.

[ANSI97] ANSI X9.63-199x. Public Key Cryptography for the Financial Services Industry: Elliptic Curve Key Agreement and Transport Protocols, 1997.

[ANSI98] ANSI X9.62-199x. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 1998.

[AR04] D. Aharonov, O. Regev. Lattice problems in NP intersect coNP. Journal of the ACM, vol. 52(5), pages 749–765, 2005.

[Bar04] M. Bardet. Etude des systèmes algèbriques surdèterminès. Applications aux codes correcteurs et à la cryptographie. PhD thesis, Universitè Paris, 2004.

[BBJ+08] D. Bernstein, P. Birkner, M. Joye, T. Lange, C. Peters. Twisted Edwards curves. Progress in Cryptology–Africacrypt 2008, Vol. 5023 of LNCS, pages 389–405, 2008.

[BCP97] W. Bosma, J. Cannon, C. Playoust. The Magma algebra system. I. The user language, Journal Symbolic Computation, vol. 24, pages 235–265, 1997.

[BFP09] L. Bettale, J. Faugère, L. Perret. Hybrid approach for solving multivariate systems over finite fields, Journal Mathematics Cryptology, vol. 3, pages 177–197, 2009.

[BFS04] M. Bardet, J. Faugère, B. Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In International Conference on Polynomial System Solving - ICPSS, pages 71–75, 2004.

[BFSY05] M. Bardet, J. Faugère, B. Salvy, B. Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. The Effective Methods in Algebraic Geometry Conference, Mega 2005, pages 1–14, 2005.

[BL07] D. Bernstein, T. Lange. Faster addition and doubling on elliptic curves. In ASIACRYPT, pages 29–50, 2007.

[BL13] D. Bernstein, T. Lange. ECRYPT Benchmarking of Cryptographic Systems. http://bench.cr.yp.to, 2013.

[BLF08] D. Bernstein, T. Lange, R. Farashahi. Binary Edwards Curves. Cryptographic Hardware and Embedded Systems–CHES 2008, Vol. 5154 of LNCS, pages 244-265, 2008.

[BK98] R. Balasubramanian, N. Koblitz. The improbability that an elliptic curve has subexponential discrete log problem under the Menezes-Okamoto-Vanstone algorithm. Journal of Cryptology, vol. 11, pages 141–145, 1998.

[Buc06] B. Buchberger. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. Journal Symbolic Computation, vol. 41(3-4), pages 475–511, 2006.

[CF05] H. Cohen, G. Frey. Handbook of elliptic and hyperelliptic curve cryptography. CRC Press, 2005.

[CH13] H. Cohn, N. Heninger. Approximate common divisors via lattices. ANTS X, vol. 1 of The Open Book Series, pages 271–293, 2013.

[Che55] C. Chevalley. Invariants of finite groups generated by reflections. American Journal of Mathematics, vol. 77(4), pages 778–782, 1955.

[CJL$^+$92] M. Coster, A. Joux, B. LaMacchia, A. Odlyzko, C. Schnorr, J. Stern. Improved low-density subset sum algorithms. Computational Complexity, vol. 2, pages 111–128, 1992.

[CKM97] S. Collart, M. Kalkbrener, D. Mall. Converting bases with the Gröbner walk. Journal of Symbolic Computation, vol. 24, pages 465–469, 1997.

[CLO07] D. Cox, J. Little, D. OShea. Ideals, Varieties, and Algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra,vol. 10, Springer, 2007.

[CMNT11] J. Coron, A. Mandal, D. Naccache, M. Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. Advances in Cryptology–CRYPTO 2011, pages 487–504, 2011.

[CN12] Y. Chen, P. Nguyen. Faster algorithms for approximate common divisors: Breaking fully homomorphic encryption challenges over the integers. Advances in Cryptology–EUROCRYPT'12, pages 502–519, 2012.

[CNT12] J. Coron, D. Naccache, M. Tibouchi. Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers. EUROCRYPT'12, vol. 7237 of LNCS, pages 446–464, 2012.

[Cop97] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. Journal of Cryptology, vol. 10(4), pages 233–260, 1997.

[Cop96a] D. Coppersmith. Finding a small root of a univariate modular equation. In EUROCRYPT, pages 155–165, 1996.

[Cop96b] D. Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In EUROCRYPT, pages 178–189, 1996.

[CP05] R. Crandall, C. Pomerance. Prime Numbers: A Computational Perspective. Second Edition, Springer, 2005.

[CS15] J. Cheon, Damien Stehlè. Fully Homomorphic Encryption over the Integers Revisited. In Proc. EUROCRYPT'15, vol. 9056-9057 of LNCS, pages 513–536, 2015.

[DG10] V. Dubois, N. Gama. The degree of regularity of HFE systems. ASIACRYPT, vol. 6477 of LNCS, pages 557–576, Springer, 2010.

[DGHV10] M. van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan. Fully homomorphic encryption over the integers. In Proc. of Eurocrypt, vol. 6110 of LNCS, pages 24–43,2010.

[DGM99] I. Duursma, P. Gaudry, F. Morain. Speeding up the discrete logarithm computation on curves with automorphisms. ASIACRYPT 1999, vol. 1716 of LNCS, pages 103–121, 1999.

[DGPS15] W. Decker, M. Greuel, M. Pfister, H. Schönemann. SINGULAR 4-0-2 — A computer algebra system for polynomial computations. http://www.singular.uni-kl.de, 2015.

[DH76] W. Diffie, M. Hellman. New directions in cryptography. IEEE Trans. Information Theory, vol. 22(6), pages 644–654, 1976.

[Die11] C. Diem. On the discrete logarithm problem in elliptic curves. Compositio Mathematica, vol. 147, pages 75–104, 2011.

[Die13] C. Diem. On the discrete logarithm problem in elliptic curves ii, Algebra and Number Theory, vol. 7, pages 1281–1323, 2013.

[DK02] H. Derksen, G. Kemper. Computational Invariant Theory. Springer Berlin Heidelberg, 2002.

[DKS98] I. Dinur, G. Kindler, S. Safra. Approximating-cvp to within almost-polynomial factors is np-hard. In Proc. of FOCS '98, pages 99–111, 1998.

[Edw07] H. Edwards. A normal form for elliptic curves. Bulletin of the American Mathematical Society vol. 44(3), pages 393–422, 2007.

[Elg85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In Advances in cryptology, vol. 196 of LNCS, pages 10–18, 1985.

[ES] N. Eén, N. Sörensson. The Minisat Page. http://www.minisat.se/.

[Fau02] J. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In Proc. of the 2002 international symposium on Symbolic and algebraic computation, ISSAC '02, pages 75–83, 2002.

[Fau99] J. Faugère. A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra, vol. 139(1-3), pages 61–88, 1999.

[FGHR13] J. Faugère, P. Gaudry, L. Huot, G. Renault. Using Symmetries in the Index Calculus for Elliptic Curves Discrete Logarithm. Journal of Cryptology, vol. 27(4), pages 595–635, 2014.

[FGLM93] J. Faugère, P. Gianni, D. Lazard, T. Mora. Efficient Computation of zero-dimensional Gröbner bases by change of ordering. Journal of Symbolic Computation, vol. 16(4), pages 329–344, 1993.

[FHJ+14] J. Faugère, L. Huot, A. Joux, G. Renault, V. Vitse. Symmetrized summation polynomials: Using small order torsion points to speed up elliptic curve index calculus. EUROCRYPT'14, vol. 8441, pages 40–57, Springer 2014.

[FJ03] J Faugère, A. Joux. Algebraic cryptanalysis of Hidden Field Equation (HFE) cryptosystems using Gröbner basis. CRYPTO, vol. 2729 of LNCS, pages 44–60, 2003.

[FNW10] R. Feng, M. Nie, H. Wu. Twisted jacobi intersections curves. Theory and Applications of Models of Computation, pages 199–210, 2010.

[FP85] U. Fincke, M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Mathematics of Computation, vol. 44(170), pages 463–471, 1985.

[FR09] J. Faugère, S. Rahmany. Solving systems of polynomial equations with symmetries using SAGBI-Gröbner basis. In Proc. of the 2009 International Symposium on Symbolic and Algebraic Computation, ISSAC'09, pages 151–158, 2009.

[FR94] G. Frey , H. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Journal Mathematics Comp., vol. 62(206), pages 865–874, 1994.

[Gal12] S. Galbraith. Mathematics of Public Key Cryptography. Cambridge University Press, Page 615, 2012.

[Gau09] P. Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. Journal of Symbolic Computation, vol. 44(12), pages 1690–1702, 2009.

[GG02] J. Gathen, J. Gerhard. Modern Computer Algebra. Cambridge University Press, 2002.

[GG14] S. Galbraith, S. Gebregiyorgis. Summation Polynomial Algorithms for Elliptic Curves in Characteristic Two, INDOCRYPT'14, vol. 8885 of LNCS, pages 409–427, 2014.

[GGH97] O. Goldreich, S. Goldwasser, S. Halevi. Eliminating decryption errors in the Ajtai-Dwork cryptosystem. In Advances in cryptology. Vol. 1294 of LNCS, pages 105–111, 1997.

[GGX11] H. Gu, D. Gu, W. Xie. Efficient Pairing Computation on Elliptic Curves in Hessian form. Information Security and Cryptology–ICISC 2010, vol. 6829 of LNCS, pages 169–176, 2011.

[GHKN06] N. Gama, N. Howgrave-Graham, H. Koy, P. Nguyen. Rankin's constant and blockwise lattice reduction. In Proc. CRYPTO'06, vol. 4117 of LNCS, pages 112–130, 2006.

[GHS02] S. Galbraith, F. Hess, N. Smart. Extending the GHS Weil descent attack. In Advances in cryptology–EUROCRYPT'02, vol. 2332 of LNCS, pages 29–44, 2002.

[Gir15] D. Giry. BlueKrypt| Cryptographic Key Length Recommendation. http://www.keylength.com/en/. BlueKrypt, vol. 29.2, 2015.

[GLV99] R. Gallant, R. Lambert, S. Vanstone. Improving the parallelized Pollard lambda search on binary anomalous curves. Mathematics of Computation, vol. 69, pages 1699–1705, 1999.

[GN08a] N. Gama, P. Nguyen. Predicting lattice reduction. In Proc. EUROCRYPT'08, vol. 4965 of LNCS, pages 31–51, 2008.

[GN08b] N. Gama and P. Nguyen. Finding short lattice vectors within Mordell's inequality. In Proc. STOC'08, pages 207–216, 2008.

[GNR10] N. Gama, P. Nguyen, O. Regev. Lattice enumeration using extreme pruning. In Proc. EUROCRYPT'10, vol. 6110 of LNCS, 2010.

[GS99] S. Galbraith, N. Smart. A cryptographic application of Weildescent. In Cryptography and coding, vol. 1746 of LNCS, pages 191–200, 1999.

[GTTD07] P. Gaudry, E. Thomé, N. Thériault, C. Diem. A double large prime variation for small genus hyperelliptic index calculus. Mathematics of Computation, vol. 76, pages 475–492, 2007.

[HG01] N. Howgrave-Graham. Approximate integer common divisors. In Cryptography and Lattices, vol. 2146 of LNCS, pages 51–66, 2001.

[HKY15] M. Huang, M. Kosters, S. Yeo. Last Fall Degree, HFE, and Weil Descent Attacks on ECDLP. Advances in Cryptology–CRYPTO 2015, vol. 9215 of LNCS, pages 581–600, 2015.

[HPST13] Y. Huang, C. Petit, N. Shinohara, T. Takagi. Improvement of Faugère et al.'s Method to Solve ECDLP. IWSEC'13, vol. 8231 LNCS, pages 115–132, 2013.

[Huf48] G. Huff. Diophantine problems in geometry and elliptic ternary forms. Duke Mathematical Journal, vol. 15, pages 443–453, 1948.

[HVM04] D. Hankerson, S. Vanstone, A. Menezes. Guide to elliptic curve cryptography. Springer Science and Business Media, 2004.

[JMS01] M. Jacobson, A. Menezes, A. Stein. Solving elliptic curve discrete logarithm problems using Weil descent. Journal Ramanujan Mathematical Society, vol. 16(3), pages 231–260, 2001.

[JP14] A. Joux, C. Pierrot. Improving the Polynomial time Precomputation of Frobenius Representation Discrete Logarithm Algorithms - Simplified Setting for Small Characteristic Finite Fields. Advances in Cryptology - ASIACRYPT 2014, vol. 8873 of LNCS, pages 378–397,2014.

[JQ01] M. Joye, J. Quisquater. Hessian elliptic curves and side channel attacks. Cryptographic Hardware and Embedded Systems–CHES 2001, vol. 2162 of LNCS, pages 402–410, 2001.

[JS98] A. Joux, J. Stern. Lattice reduction: A toolbox for the cryptanalyst. Journal Cryptology, vol. 11(3), pages 161–185, 1998.

[JTV10] M.Joye, M. Tibbouchi, D. Vergnaud. Huff's Model for Elliptic Curves. Algorithmic Number Theory–ANTS-IX, vol. 6197 of LNCS, pages 234–250, 2010.

[JV13] A. Joux, V. Vitse. Elliptic curve discrete logarithm problem over small degree extension fields - application to the static Diffie-Hellman problem on $E(\mathbb{F}_{q^5})$. Journal Cryptology, vol. 26(1), pages 119–143, 2013.

[Kan01] R. Kane. Reflection Groups and Invariant Theory. Springer, 2001.

[Kar15] K. Karabina. Point decomposition problem in binary elliptic curves. Cryptology ePrint Archive, 2015/319, 2015.

[Kem96] G. Kemper. Calculating Invariant Rings of Finite Groups over Arbitrary Fields. Journal of Symbolic Computation, vol. 21(3), pages 351–366, 1996.

[Kho04] S. Khot. Hardness of approximating the shortest vector problem in lattices. In Proc. 45th Annual IEEE Symp. on Foundations of Computer Science, pages 126–135, 2004.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. Mathematics of Computation, vol. 48(177), pages 203–209, 1987.

[Kob98] N. Koblitz. An Elliptic Curve Implementation of the Finite Field Digital Signature Algorithm. Advances in Cryptology–CRYPTO'98, vol. 1462 of LNCS, pages 327–337, 1998.

[Lan01] E. Landquist. MATH 488: Cryptographic Algorithm. http://www.cs.virginia.edu/crab/QFS_Simple.pdf, 2001.

[Laz83] D. Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In Proc. of the European Computer Algebra Conference on Computer Algebra, vol. 162 of LNCS, 1983.

[Len87] H. Lenstra. Factoring integers with elliptic curves. Annuals of Mathematics, vol. 126, pages 649–673, 1987.

[Len93] A. Lenstra. The Development of the Number Field Sieve. LNM, vol. 1554, Springer, 1993.

[LLL82] A. Lenstra, H. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. Mathematische Annalen, vol. 261(4), pages 515–534, 1982.

[LO85] J. Lagarias, A. Odlyzko. Solving low-density subset sum problems. Journal of the Association for Computing Machinery, 1985.

[LS01] P. Liardet, N. Smart. Preventing SPA/DPA in ECC systems using the Jacobi form. Cryptographic Hardware and Embedded Systems–CHES 2001, vol. 2162 of LNCS, pages 391–401, 2001.

[LV01] A. Lenstra, E. Verheul. Selecting cryptographic key sizes. Journal of Cryptology, vol. 14(4), pages 255–293, 2001.

[Mac16] F. Macaulay. The algebraic theory of modular systems. Cambridge University Press, vol. xiv(19), page 112, 1916.

[Mac27] F.S. Macaulay. Some properties of enumeration in the theory of modular systems. Proc. London Mathematical Society, vol. 26, pages 531–555, 1927.

[MCP07] C. McDonald, C. Charnes, J. Pieprzyk. Attacking Bivium with MiniSat. ECRYPT Stream Cipher Project, Report 2007/040, 2007.

[Men93] A. Menezes. Elliptic Curve Public Key Cryptosystems. Kluwer Academic Publishers, 1993.

[MH78] R. Merkle, M. Hellman. Hiding Information and Signatures in Trapdoor Knapsacks. IEEE Transactions on Information Theory, vol. 24, pages 525–530, 1978.

[Mic98] D. Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. FOCS'98, pages 92–98, 1998.

[Mil04] V. Miller. The Weil pairing, and its efficient calculation. Journal of Cryptology, vol. 17, pages 235–261, 2004.

[Mil86] V. Miller. Use of elliptic curves in cryptography. Advances in cryptology–CRYPTO 85, vol. 218 of LNCS, pages 417–426, 1986.

[Mon87] P. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. Mathematics of computation, vol. 48(177), pages 243–264, 1987.

[MOV93] A. Menezes, T. Okamoto, S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. Transactions on Information Theory, vol. 39(5), pages 1639–1646, 1993.

[MV10] D. Micciancio, P. Voulgaris. Faster exponential time algorithms for the shortest vector problem. In Proc. SODA'10, pages 1468–1480, 2010.

[Nag13] K. Nagao. Decomposition formula of the Jacobian group of plane curve. Cryptology ePrint Archive, Report 2013/548, 2013.

[Nak09] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf, 2009.

[NIST94] National Institute for Standards and Technology. Digital signature standard, FIPS PUB 186, 1994.

[NS01] P. Nguyen, J. Stern. The two faces of lattices in cryptology. CaLC'01, vol. 2146 of LNCS, pages 146–180, Springer, 2001.

[DT14] J. Ding, C. Tao. A New Algorithm for Solving the General Approximate Common Divisors Problem and Cryptanalysis of the FHE Based on the GACD problem. Cryptology ePrint Archive, Report 2014/042, 2014.

[NV08] P. Nguyen, T. Vidick. Sieve algorithms for the shortest vector problem are practical. Journal of Mathematical Cryptology, vol. 2(2), pages 181–207, 2008.

[OM99] P. van Oorschot, M. Wiener. Parallel collision search with cryptanalytic applications. Journal of Cryptology, vol. 12, pages 1–28, 1999.

[PH78] S. Pohlig, M. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. IEEE Transactions on Information Theory, vol. IT-24, pages 106–110, 1978.

[Poh81] M. Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. SIGSAM Bulletin, vol. 15(1), pages 37–44, 1981.

[Pol74] J. Pollard. Theorems on factorisation and primality testing. Proc. of the Cambridge Philosophical Society, vol. 76, pages 521–528, 1974.

[Pol78] J. Pollard. Monte Carlo methods for index computation (mod p). Mathematical Computation, vol. 32(143), pages 918–924, 1978.

[PQ12] C. Petit, J. Quisquater. On Polynomial Systems Arising from a Weil Descent. ASIACRYPT'12, vol. 7658, pages 451–466, 2012.

[PTH15] C. Petit, T. Takagi, Y. Huang. On generalised first fall degree assumptions, Cryptology ePrint Archive, Report 2015/358, 2015.

[Reg04] O. Regev. New lattice-based cryptographic constructions. Journal of the ACM, vol. 51(6), pages 899–942, 2004.

[RSA78] R. Rivest, A. Shamir, L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, vol. 21(2), pages 120–126, 1978.

[SA98] T. Satoh, K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. Comment. Math. Univ. St. Paul., vol. 47(1), pages 81–92, 1998.

[SAT$^+$98] C. Gomes, B. Selman, H. Kautz. Boosting combinatorial search through randomization. AAAI, pages 431–437, 1998.

[Sch87] C. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. Theoretical Computer Science, vol. 53(2-3), pages 201–224, 1987.

[SE05] N. Sörensson, N. Eén. MiniSat-A SAT Solver with Conflict-Clause Minimization. SAT'05, 2005.

[SE08] N. Sörensson, N. Eén. Minisat 2.1 and minisat++ 1.0 sat race 2008 editions, 2008.

[SE91] C. Schnorr, M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. FCT'91, vol. 529 of LNCS, pages 68–85, Springer, 1991.

[SE94] C. Schnorr, M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming, vol. 66, pages 181–199, 1994.

[Sem04] I. Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2004/031, 2004.

[Sem15] I. Semaev. New algorithm for the discrete logarithm problem on elliptic curves, Cryptology ePrint Archive, Report 2015/310, 2015.

[Sem98] I. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. Mathematical Computation, vol. 67(221), pages 353–356, 1998.

[Sha71] D. Shanks. Class number, a theory of factorization, and genera. Proc. Sympos. Pure Math., vol. 20, pages 415–440, 1971.

[Sho97] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM, vol. 26(5), pages 1484–1509, 1997.

[Sil09] J. Silverman. The Arithmetic of Elliptic Curves. GTM vol. 106, 1986. Expanded 2nd Edition, 2009.

[Sma99] N.P. Smart. The discrete logarithm problem on elliptic curves of trace one. Journal Cryptology, vol. 12(3), pages 193–196, 1999.

[ST13] M. Shantz, E. Teske. Solving the Elliptic Curve Discrete Logarithm Problem Using Semaev Polynomials, Weil Descent and Gröbner Basis Methods-An Experimental Study. Number Theory and Cryptography, vol. 8260, pages 94–107, 2013.

[ST54] G. Shephard, J. Todd. Finite unitary reflection groups. Canadian Journal of Mathematics, vol. 6, pages 274–304, 1954.

[Sti56] D. Stinson. Cryptography : theory and practice. CRC Press, 1995.

[Thé03] N. Thériault. Index calculus attack for hyperelliptic curves of small genus. In Advances in Cryptology–ASIACRYPT'03, vol. 2894 of LNCS, pages 75–92, 2003.

[TK13] A. Takayasu, N. Kunihiro. Better Lattice Constructions for Solving Multivariate Linear Equations Modulo Unknown Divisors. ACISP'13, 2013.

[Wal78] R. Walker. Algebraic Curves. Springer, 1978.

[Wei49] A. Weil. Numbers of solutions of equations in finite fields. Bulletin of the American Mathathematical Society, vol. 55(5), pages 497–508, 1949.

[Wie86] D. Wiedemann. Solving sparse linear equations over finite fields. IEEE Transactions on Information Theory, vol. 32(1), pages 54–62, 1986.

[WZ98] M. Wiener, R. Zuccherato. Faster attacks on elliptic curve cryptosystems. Selected Areas in Cryptography'98, vol. 1556 of LNCS, pages 190–120, 1998.

[YC04] B. Yang, J. Chen. Theoretical analysis of XL over small fields. ACISP, vol. 3108, pages 277–288, 2004.

[YCC04] B. Yang, J. Chen, N. Courtois. On asymptotic security estimates in XL and gröbner bases-related algebraic cryptanalysis. ICICS, vol. 3269, pages 401–413, 2004.