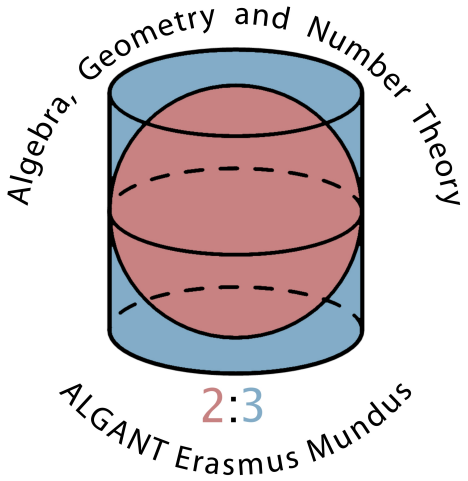


Algant Master Thesis

Tower decomposition of Hilbert class fields

Candidate:
Jared Guissmo Asuncion

Advisor:
Dr. Andreas Enge

Universiteit Leiden

Universiteit Leiden

université de BORDEAUX

Université de Bordeaux

July 2016

Contents

Chapter 1

Introduction

One of the more recently discovered primality proving algorithms was the Atkin-Morain elliptic curve primality test [1]. One step of this algorithm requires the construction of an elliptic curve over a finite field $\mathbb{F}_p$, where p is the integer which we want to prove prime. We want such elliptic curve to have exactly N points, where N is an integer with a large prime factor. The best known method to find such a curve is by computing an invariant, called the j-invariant, which we can use to produce such an elliptic curve. We write the algorithm here [3] for reference.

Algorithm 1.1. `CMalgo`

INPUT: an integer $N > 6$ and a prime p such that $|N + 1 - p| \leq 2\sqrt{N}$

OUTPUT: an elliptic curve over $E/\mathbb{F}_p$ with $|E(\mathbb{F}_p)| = N$.

ALGORITHM:

1. Compute the Hilbert class polynomial $h_K \in \mathbb{Z}[X]$ for the field $K = \mathbb{Q}(\sqrt{-D})$ where $-D = (p+1-N)^2 - 4p$.
2. Compute a root $j \in \mathbb{F}_p$ of h_K, viewed as a polynomial over $\mathbb{F}_p$.
3. Let
$$E = \begin{cases} Y^2 = X^3 + aX - a & \text{for } j \neq 0, 1728 \\ Y^2 = X^3 + 1 & \text{for } j = 0 \\ Y^2 = X^3 + X & \text{for } j = 1728 \end{cases}$$
where $a = \frac{27j}{4(1728-j))}$.
4. If $|E(\mathbb{F}_p)| = N$, return E. Otherwise, return its quadratic twist.

The most important step in algorithm 1.1 is to find a root $j \in \mathbb{F}_p$ of the Hilbert class polynomial h_K, a polynomial of degree h which is defined in section 2.2. This polynomial generates the Hilbert class field, H_K of K. It is defined in section 2.1 to be the maximal unramified abelian extension of K. By taking the intermediate fields of H_K/K and obtaining their respective defining polynomials, we only need to find roots of polynomials of lesser degree. This thesis aims to provide a more thorough description of the algorithm in [6] which gives a tower of intermediate fields and their respective defining polynomials to aid in the computation of algorithm 1.1. We state the more general problem below.

Problem 1.2. Given a finite abelian extension M/K generated by the polynomial h, find an intermediate field L and the respective polynomials W and V, for M/L and L/K.

Note that we can let $M = H_K$, and we can recursively apply the algorithm we obtain from solving problem 1.2 to make a finer decomposition. Since M/K is a Galois extension, we can find an intermediate field L by taking a normal subgroup H of the Galois group $\mathrm{Gal}(M/K)$. However, in general, elements of G are difficult to work with symbolically. And so, we will prefer to work with the more convenient form class group discussed in section 2.3. From here, it turns out that we need not compute the Hilbert class polynomial as we can compute the roots from the form class group. Not only that, but in section 2.4, we see that we need not compute the roots of the Hilbert class polynomial. There can exist other minimal polynomials which generate the Hilbert class field and these are the ones we work with in a running example scattered around the different sections which cumulates in the big example 4.20.

Moreover, we can avoid doing intermediate computations with complex numbers and compute minimal polynomials in real subfields of M, L and K instead. This is because the minimal polynomials we get are polynomials in $\mathbb{Z}$. However, it turns out that these real subfields M', L' and K' are not necessarily Galois, unlike their counterparts. On one hand, computations are faster with the subfields but we also want to utilise the Galois group offered by the original fields. It turns out we can get the best of both worlds as explained in section 3.1. In the following section 3.2, we discuss algorithms to quickly compute certain polynomials given their roots.

Finally, we build algorithm 1.2 to solve problem 4.19. In theory, this solves the problem of finding a tower of intermediate fields of maximal length. We treat the algorithm in [6] more thoroughly in this thesis and discuss in more detail some tricks on how to efficiently transition from one iteration to the next. Some strategies are discussed in sections 4.1 and 4.2 on how to save computations between iterations. We conclude that part by stating the main algorithm, algorithm 4.19.

We also analyze in detail the time complexity of the algorithm in section 5. In section 6, we briefly deal with the inverse problem to provide a way to check if our computations and/or implementations are correct. Some statistics with regards to the actual running time can be found in chapter 7.

Chapter 2

Generating the Hilbert Class Field

2.1 The Hilbert Class Field

In this section, we define the Hilbert class field and its relation with the ideal class group defined as follows:

Definition 2.1. Let K be an imaginary quadratic number field. Let $\mathcal{O}_K$ be its ring of integers. We define the ideal class group of K to be

$$\mathrm{Cl}(\mathcal{O}_K) = \frac{\mathcal{I}(\mathcal{O}_K)}{\mathcal{P}(\mathcal{O}_K)} = \frac{\text{fractional ideals of } \mathcal{O}_K}{\text{principal fractional ideals of } \mathcal{O}_K}.$$

Before proceeding to the definition of the Hilbert class field, we recall first what it means for a field extension to be unramified and abelian. Similarly, if L is an extension of K, we can also speak of $\mathcal{O}_L$. A prime $\mathfrak{p} \in \mathcal{O}_K$ is said to be unramified if its prime decomposition in $\mathcal{O}_L$ is squarefree, that is,

$$\mathfrak{p}\mathcal{O}_K = \mathfrak{P}_1\mathfrak{P}_2 \cdots \mathfrak{P}_g$$

where the $\mathfrak{P}_i$ are distinct. From these definitions, we are now ready to define what it means for a field extension to be unramified.

Definition 2.2. Let $K = \mathbb{Q}(\sqrt{-D})$ be an imaginary quadratic number field. An extension L/K is unramified if all prime ideals in $\mathcal{O}_K$ are unramified.

In general, there is also a notion of unramified (infinite) places which corresponding to the embeddings $K \hookrightarrow \mathbb{C}$. However, we need not worry about these embeddings in the particular case where K is an imaginary quadratic number field. This is due to the fact that all embeddings of K are already complex, meaning that given any field extension, L, the infinite places corresponding to the complex embeddings will always be unramified. Now, we define what it means to be an abelian extension.

Definition 2.3. An extension L/K is abelian if it is Galois with abelian Galois group.

Let L/K be a finite unramified abelian extension. Let $\mathfrak{p}$ be a prime ideal of $\mathcal{O}_K$ and $\mathfrak{P}$ be a prime ideal of $\mathcal{O}_L$ above $\mathfrak{p}$. We know that the extension $l/k = \frac{\mathcal{O}_L/\mathfrak{P}}{\mathcal{O}_K/\mathfrak{p}}$ is cyclic and is generated by the Frobenius automorphism $\mathrm{Frob}_\mathfrak{p}$ such that $\mathrm{Frob}_\mathfrak{p}(x) = x^{N(\mathfrak{p})}$. Since $\mathfrak{P}$ is unramified, there exists a unique $\sigma_\mathfrak{p}$ such that $\sigma_\mathfrak{p}(x) \equiv x^{N(\mathfrak{p})} \pmod{\mathfrak{P}}$. We call $\sigma_\mathfrak{p}$ the Artin symbol for $\mathfrak{p}$. Hence, we can define the Artin map

$$\left(\frac{\cdot}{L/K}\right) : \mathcal{I}(\mathcal{O}_K) \to \mathrm{Gal}(L/K)$$

$$\mathfrak{p}_1^{e_1} \cdots \mathfrak{p}_t^{e_t} \mapsto \sigma_{\mathfrak{p}_1}^{e_1} \cdots \sigma_{\mathfrak{p}_t}^{e_t}$$

Class field theory tells us that there exists a maximal unramified abelian extension H_K of K such that the Artin map induces an isomorphism

$$\mathrm{Cl}(\mathcal{O}_K) = \frac{\mathcal{I}(\mathcal{O}_K)}{\mathcal{P}(\mathcal{O}_K)} \cong \mathrm{Gal}(H_K/K).$$

This extension H_K is unique and it is called the Hilbert class field of K. In the next part, we find that for imaginary quadratic fields K such that $\tau_2/\tau_1 \notin \mathbb{R}$ it is generated by the value of a modular function which depends on the ideal class group.

2.2 The Hilbert Class Polynomial

Recall that a lattice of full rank Λ is an additive subgroup of $\mathbb{C}$ with a $\mathbb{Z}$-basis τ_1 and τ_2. We write $\Lambda = [\tau_1, \tau_2] = \mathbb{Z}\tau_1 + \mathbb{Z}\tau_2$. Without loss of generality, assume $\mathrm{Im}(\frac{\tau_1}{\tau_2}) > 0$ (otherwise, switch τ_1 and τ_2). The j-invariant of a lattice Λ is defined to be

$$j(\Lambda) = 1728 \cdot \frac{g_2(\Lambda)^3}{g_2(\Lambda)^3 - 27g_3^2(\Lambda)}$$

where

$$g_2(\Lambda) := 60 \sum_{\tau \in \Lambda \setminus \{0\}} \frac{1}{\tau^4} \qquad \text{and} \qquad g_3(\Lambda) := 140 \sum_{\tau \in \Lambda \setminus \{0\}} \frac{1}{\tau^6}.$$

Note that we can also define j as a function from $\mathbb{C} \to \mathbb{C}$ by taking $\tau := \frac{\tau_1}{\tau_2}$ and defining

$$j(\tau) := j(\Lambda).$$

Since j only depends on the lattice, then j is invariant under any unimodular transformation, i.e. transformations of the form

$$\tau \mapsto \frac{a\tau + b}{c\tau + d}$$

where $a, b, c, d \in \mathbb{Z}$ and $ad - bc = 1$. From here, we can also define j as a function from the ideal class group $\mathrm{Cl}(\mathcal{O}_K)$. Let $\mathfrak{k} \in \mathrm{Cl}(\mathcal{O}_K)$. Take $I = [\tau_1, \tau_2]$ be the $\mathbb{Z}$-basis of an ideal which is a representative of the class $\mathfrak{k}$. Let $\tau = \frac{\tau_1}{\tau_2}$ and let

$$j(\mathfrak{k}) = j(I) = j\left(\frac{\alpha_1}{\alpha_2}\right).$$

Note that this is well-defined since any other representative I' will be of the form $I' = (\alpha)I = (\alpha\tau_1, \alpha\tau_2)$ and will result in the same τ. Moreover, it can be shown that $j(\mathfrak{k})$ is an algebraic integer. Furthermore, we have the theorem from [8]:

Theorem 2.4. Let K be an imaginary quadratic number field. For every ideal class $\mathfrak{k} \in \mathrm{Cl}(\mathcal{O}_K)$, we have

$$H_K = K(j(\mathfrak{k})).$$

Moreover, there exists an isomorphism via the following well-defined map

$$\begin{aligned} \sigma : \mathrm{Cl}(\mathcal{O}_K) &\to \mathrm{Gal}(H_K/K) \\ \mathfrak{k} &\to \sigma_{\mathfrak{k}}. \end{aligned} \tag{2.5}$$

such that if $\mathfrak{p}$ is a representative of $\mathfrak{k}$, then $\sigma_{\mathfrak{k}}$ is the Frobenius automoprhism associated with $\mathfrak{p}$. Furthermore,

$$j(\mathfrak{k})^{\sigma_{\mathfrak{h}}} = j(\mathfrak{k}\mathfrak{h}^{-1})$$

and

$$\overline{j(\mathfrak{h})} = j(\mathfrak{h}^{-1})$$

for all $\mathfrak{h} \in \mathrm{Cl}(\mathcal{O}_K)$

Finally, we define the Hilbert class polynomial to be the minimal polynomial of any of the $j(\mathfrak{k})$. We define

$$h_K(X) = \prod_{\sigma \in \mathrm{Gal}(H_K/K)} (X - x^\sigma) = \prod_{\mathfrak{h} \in \mathrm{Cl}(\mathcal{O}_K)} (X - x^{\sigma_\mathfrak{h}})$$

where $x := j(\mathfrak{k})$ for some fixed $\mathfrak{k}$. From this, we can define a group structure on the roots of h_K by defining

$$x_0 x_1 = x^{\sigma_0} x^{\sigma_1} := x^{\sigma_0 \sigma_1}.$$

2.3 The Form Class Group

The form class group is yet another group in our growing list of objects that are isomorphic to each other. Compared to all the previous groups we have defined, the form class group is very easy to represent. We start by defining a binary quadratic form. A binary quadratic form of discriminant $-D$ is a homogeneous polynomial

$$Q(x, y) = Ax^2 + Bxy + Cy^2 = [A, B, C]$$

such that its discriminant is $-D = \sqrt{B^2 - 4AC}$. If $\gcd(A, B, C) = 1$, then it is said to be primitive. If $q(x, y) > 0$ whenever $(x, y) \neq (0, 0)$ then it is said to be positive definite. Moreover quadratic forms q_1 and q_2 are said to be equivalent (written $q_1 \sim q_2$) if there exists $a, b, c, d \in \mathbb{Z}$ where $ad - bc = 1$ such that

$$q_1(x, y) = q_2(ax + by, cx + dy).$$

A primitive positive definite binary quadratic form is said to be reduced if

$$|B| \leq A \leq C \qquad \text{and} \qquad B \geq 0 \text{ if either } |B| = A \text{ or } A = C.$$

Every primitive positive definite binary quadratic form is equivalent to a unique reduced form. Hence, we can talk of the set $\mathrm{Cl}(-D)$ of classes of forms of discriminant $-D$. We can then define a bijection

$$\begin{aligned} \Theta : \mathrm{Cl}(-D) &\to \mathrm{Cl}(\mathcal{O}_K) \\ [A, B, C] &\mapsto \left(A, \frac{-B + \sqrt{-D}}{2}\right) \end{aligned} \tag{2.6}$$

Moreover, this bijection also gives a group structure to $\mathrm{Cl}(-D)$ and so from now on, we refer to it as the form class group. A detailed description of this group is on [4].

2.4 Class Invariants

From our previous discussion, we can now compute all the roots of the Hilbert class polynomial, h_K, using only the reduced forms of discriminant $-D$. This is because each reduced form $[A, B, C]$ is associated to one of the equivalence classes in $\mathrm{Cl}(-D)$. Using the isomorphism Θ from (2.6), we have $\mathfrak{h} = \Theta([A, B, C]) = \left(A, \frac{-B+\sqrt{-D}}{2}\right)$ is one of the elements in the ideal class group of K where $K = \mathbb{Q}(\sqrt{-D})$. Applying the isomorphism σ from

(2.5) to $\mathfrak{h}$, we obtain $\sigma_{\mathfrak{h}}$. Fixing $\mathfrak{k} \in \mathrm{Cl}(\mathcal{O}_K)$ to be the trivial class, we let $x = j(\mathfrak{k})$. And so, by theorem 2.4, $(j(\mathfrak{h}))_{\mathfrak{h}\in\mathrm{Cl}(\mathcal{O}_K)}$ is a complete set of conjugates of x. Thus,

$$h_K(X) = \prod_{\mathfrak{h}\in\mathrm{Cl}(\mathcal{O}_K)} (X - j(\mathfrak{h})) = \prod_{[A,B,C]\in\mathrm{Cl}(-D)} \left(X - j\left(\frac{-B+\sqrt{-D}}{2A}\right)\right)$$

because of these chain of isomorphisms (2.7)

$$\begin{array}{ccccccc} \mathrm{Cl}(-D) & \xleftrightarrow{\sim} & \mathrm{Cl}(\mathcal{O}_K) & \xleftrightarrow{\sim} & \mathrm{Gal}(H_K/K) & \xleftrightarrow{\sim} & \text{roots of } h_K \\ [A,B,C] & \xmapsto{\Theta} & \mathfrak{h} = \left[\left(A, \frac{-B+\sqrt{-D}}{2}\right)\right] & \xmapsto{\sigma} & \sigma_{\mathfrak{h}^{-1}} & \xmapsto{x^-} & x^\sigma = j(\mathfrak{k}\mathfrak{h}^{-1}) = j\left(\frac{-B-\sqrt{-D}}{2A}\right). \end{array}$$

And so, we are now able to find the polynomial which generates the Hilbert class field. However, this polynomial involves very large coefficients. For example, for $K = \mathbb{Q}(\sqrt{-455})$, we have that

$$\begin{aligned} h_K(X) = {} & X^{20} + 126806719749729443341753846625X^{19} - 6336048071569347919137157351602155143 75X^{18} \\ & + 160799441768696845745670662555481206342004696815242983437 50X^{17} - \dots \\ & - 321818849693382992374572733241808203312239458035475891239018393779776371040\dots. \end{aligned}$$

Observe that the 142-digit constant term will not even fit the space. Luckily, there are a lot of other choices for the function j.

Definition 2.8. For a modular function u, its value $u(\tau)$ is called a class invariant if $u(\tau)$ is in $K(j(\tau)) = H_K$.

For example, we take from [1] the following theorem

Theorem 2.9. Let $D \not\equiv 0 \pmod 3$. Suppose $AX^2 + 2BX + C = 0$ with $4B^2 - 4AC = -4D$ and satisfies either of the following:

- A and C are odd, $3|B$ (2.10)
- $A \equiv C \equiv 0 \pmod 3$ and $B \not\equiv 0 \pmod 3$. (2.11)

Then letting $(2/A)$ be the Kronecker symbol (i.e. $(2/A) = 1$ if 2 is a square mod A and -1 otherwise), the following are true:

- If $D \equiv 1 \pmod 8$ and $B \equiv 2((2/A) - 1) \pmod 8$, then $f(\tau)^2/\sqrt{2}$ is a class invariant.
- If $D \equiv 5 \pmod 8$ and $B \equiv 2((2/A) - 1) \pmod 8$, then $f(\tau)^4$ is a class invariant.
- If $D \equiv \pm 2 \pmod 8$ and $B \equiv 2((2/A) - 1) \pmod 8$, then $f_1(\tau)/\sqrt{2}$ is a class invariant.
- If $D \equiv 3 \pmod 8$ and $B \equiv 4((2/A) - 1) \pmod{16}$, then $f(\tau)$ is a class invariant.
- If $D \equiv 7 \pmod 8$ and $B \equiv 4((2/A) - 1) \pmod{16}$, then $f(\tau)/\sqrt{2}$ is a class invariant.

where

$$f(z) = e^{-i\pi/24} \cdot \frac{\eta(\frac{z+1}{2})}{\eta(z)}, \qquad f_1(z) = \frac{\eta(\frac{z}{2})}{\eta(\tau)}, \qquad f_2(z) = \frac{\eta(2z)}{\eta(z)}$$

and

$$\eta(z) = q^{1/24} \cdot \prod_{n=1}^{\infty}(1 - q^n) \qquad \text{with } q = e^{2\pi i\omega}.$$

Definition 2.12. Let $D \not\equiv 0 \pmod 3$ and $D \equiv 0 \pmod 4$.

$$H_D[u](X) = \prod_{Q \in \mathrm{Cl}(-D)} (X - u(\tau_Q))$$

where

- u is the corresponding class invariant as in theorem 2.9
- $Q' = [A, B, C]$ is a representative of the form class Q satisfying (2.10) and (2.11)
- τ_Q satisfies the polynomial $Az^2 + 2Bz + C = 0$ with discriminant D.

We then state without proof the following theorem.

Theorem 2.13. Let $-D$ be a fundamental discriminant such that $D \not\equiv 0 \pmod 3$.

- If $D \equiv 0 \pmod 4$, then $H_D[u](X)$ generates the Hilbert class field H_K of $K = \mathbb{Q}(\sqrt{-D})$.
- If $D \equiv 3 \pmod 4$, then $H_{4D}[u](X)$ generates the Hilbert class field H_K of $K = \mathbb{Q}(\sqrt{-D})$. In this case $\mathrm{Cl}(-D) = \mathrm{Cl}(-4D)$.

In general, the polynomials in 2.13 have smaller coefficients than the Hilbert class polynomials. This is illustrated in the following example.

Example 2.14. Let $K = \mathbb{Q}(\sqrt{-D})$ with $D = 455 \equiv 3 \pmod 4$. Using an algorithm discussed in [1], for each class in $\mathrm{Cl}(-4D)$, we produce a representative $Q' = [A, B, C]$ which satisfy either (2.10) or (2.11). Still following theorem 2.9, we obtain $u(\tau) = f(\tau)/\sqrt{2}$. We make a table showing the calculations. We also devote a column for the reduced form Q corresponding to each Q'. Moreover, take note that $\tau_Q = \frac{-B' + \sqrt{-4D}}{2A'}$ where $Q' = [A', B', C']$.

Q	Q'	$u(\tau_Q)$	Q	Q'	$u(\tau_Q)$
$[1, 0, 455]$	$[1, 0, 455]$	11.54	$[5, 0, 91]$	$[5, 240, 2971]$	-1.236
$[3, 2, 152]$	$[3, 176, 2733]$	$-1.374 - 1.153i$	$[15, -10, 32]$	$[15, 320, 1737]$	$0.1554 + 0.8326i$
$[9, -4, 51]$	$[9, 320, 2895]$	$-0.6623 + 0.7015i$	$[19, 2, 24]$	$[19, 1104, 16061]$	$-0.6251 - 0.4916i$
$[17, 4, 27]$	$[17, 480, 3415]$	$-0.2266 + 0.8176i$	$[8, -2, 57]$	$[57, 2624, 30207]$	$-0.4952 + 0.04877i$
$[16, -6, 29]$	$[29, 528, 2419]$	$0.2375 + 0.6643i$	$[21, -14, 24]$	$[21, 112, 171]$	$0.7532 + 0.2435i$
$[13, 0, 35]$	$[13, 624, 7523]$	-0.8816	$[7, 0, 65]$	$[7, 0, 65]$	1.054
$[16, 6, 29]$	$[29, 2256, 43891]$	$0.2375 - 0.6643i$	$[21, 14, 24]$	$[21, 1904, 43179]$	$0.7532 - 0.2435i$
$[17, -4, 27]$	$[17, 1152, 19543]$	$-0.2266 - 0.8176i$	$[8, 2, 57]$	$[57, 2848, 35583]$	$-0.4952 - 0.04877i$
$[9, 4, 51]$	$[9, 544, 8271]$	$-0.6623 - 0.7015i$	$[19, -2, 24]$	$[19, 720, 6845]$	$-0.6251 + 0.4916i$
$[3, -2, 152]$	$[3, 112, 1197]$	$-1.374 + 1.153i$	$[15, 10, 32]$	$[15, 1120, 20937]$	$0.1554 - 0.8326i$

Taking the polynomial which has all the $u(\tau_{Q'})$ as the roots, we obtain the polynomial

$$\begin{aligned} H_{-4D}[f/\sqrt{2}](X) = {} & X^{20} - 6X^{19} - 50X^{18} - 142X^{17} - 200X^{16} - 129X^{15} + 38X^{14} + 191X^{13} + 246X^{12} \\ & + 194X^{11} + 76X^{10} - 30X^9 - 73X^8 - 57X^7 - 15X^6 + 16X^5 + 26X^4 + 23X^3 + 15X^2 + 6X + 1 \end{aligned} \tag{2.15}$$

which generates the extension H_K/K.

Definition 2.16. Let $Q = [A, B, C]$ be a quadratic form of discriminant $-D$ and let the associated automorphism (via the isomorphism in (2.7)) $\sigma_Q \in \mathrm{Gal}(H_K/K)$ act on $x = \tau(\tilde{Q})$ for some fixed $\tilde{Q} \in \mathrm{Cl}(-D)$ as follows:

$$x^{\sigma_Q} = u(\tau_{Q^{-1}\tilde{Q}})$$

Moreover, we let complex conjugation act on x as follows:

$$\overline{u(\tau_Q)} = u(\tau_Q^{-1}). \tag{2.17}$$

Remark 2.18. $u(Q)$ is real if and only if Q has at most order 2.

Chapter 3

Theory for the Algorithm

In this chapter, we lay down all the basic theory we need for the main algorithm 4.19. We start 3.1 by producing an intermediate field L in the case that M/K is a Galois extension. After tackling this case, we move forward and discuss the non-Galois case discussed in [6]. In section 3.2, we define a way to symbolically express an element of L from a floating point representation.

3.1 Galois Theory

3.1.1 Galois Case

Let M be a Galois extension of K with Galois group $G = \mathrm{Gal}(M/K)$. Fix $x \in M$ to be one of the roots of the defining polynomial f. Thus, we can write

$$f(X) = \prod_{g \in G} (X - x^g) \tag{3.1}$$

where x^g denotes the image of x under the automorphism $g : M \to M$. Let H be a normal subgroup of G. By the fundamental theorem of Galois theory, there exists a corresponding fixed field

$$L = M^H = \{y \in M : y^\sigma = y \text{ for all } \sigma \in H\}$$

such that the degree $[M : L]$ of the extension M/L is equal to $|H|$. Now, define

$$W(X) = \prod_{h \in H} (X - x^h). \tag{3.2}$$

First, we show the following theorem.

Theorem 3.3. W is a polynomial in $L[X]$ and it is irreducible.

Proof. Let $h' \in H$. Then

$$W(X)^{h'} = \prod_{h \in H} (X - (x^h)^{h'}) = \prod_{h \in H} (X - x^{h'h}) = \prod_{\tilde{h} \in h'H} (X - x^{\tilde{h}}) = W(X)$$

where the last equality is because $h'H = H$. Thus, W is fixed by H and hence $W \in L[X]$. Moreover, x is a root of U and the degree $|H|$ of W is equal to the degree of the field extension M/L. Since $M = L(x)$, we conclude that W is an irreducible polynomial over L. □

Hence, W is a minimal polynomial of x over the field L and we have the following.

Corollary 3.4. *W is a defining polynomial of the extension M/L*

Now, for $g \in G$, define

$$U_g(X) = \prod_{h \in H} (X - x^{hg}).$$

Theorem 3.5. We have the following properties for U_g.

1. If $g, g' \in G$ then
$$U_g(X)^{g'} = U_{g'g}(X) \tag{3.6}$$
2. If $h \in H$, then
$$U_g(X)^h = U_g(X) \tag{3.7}$$
3. If $g' \in Hg$, then
$$U_{g'}(X) = U_g(X) \tag{3.8}$$

Proof. For (3.6), if $g, g' \in G$, we have

$$U_g(X)^{g'} = \prod_{h \in H} (X - (x^{hg})^{g'}) = \prod_{h \in H} (X - (x^{g'hg})) = \prod_{h \in H} (X - (x^{g'hg'^{-1}g'g})) = \prod_{h' \in g'Hg'^{-1}} (X - (x^{h'g'g})) = U_{g'g}(X).$$

where the last equality is due to the fact that H is a normal subgroup, and thus $g'Hg'^{-1} = H$. For equation (3.7), if $h' \in H$, then

$$U_g(X)^h = \prod_{h \in H} (X - x^{hh'}) = \prod_{\tilde{h} \in Hh'} (X - x^{\tilde{h}}) = U_g(X)$$

where the last equality is due to the fact that H is a group. For (3.8), suppose $g' \in Hg$. Then $g' = h'g$ for some $h' \in H$. Thus, $g'g^{-1} = h'$. And so, $U_{g'} = U_{g'g^{-1}g} = U_{h'g} = U_g^{h'} = U_g$. □

Hence, we can define

$$U_{gH} = U_g \tag{3.9}$$

for every $gH \in G/H$. And this is well-defined by theorem 3.5. Finally, we note that

$$f(X) = \prod_{gH \in G/H} U_{gH}.$$

Moreover, we remark that $W = U_{1_G}$ where 1_G is the identity in G. Let $m = |H|$. Now, we write

$$W = U_{1_G} = X^m + \sum_{j=0}^{m-1} (-1)^{m-j} \vartheta_j X^j. \tag{3.10}$$

Moreover, note that for each $g \in G/H$

$$U_g = X^m + \sum_{j=0}^{m-1} (-1)^{m-j} \vartheta_j^g X^j.$$

Just as in the case of M/L, we wish to find a defining polynomial for L/K. We define for all $j = 1, \ldots, m-1$:

$$V_j(Y) = \prod_{g \in G/H} (Y - \vartheta_j^g). \tag{3.11}$$

Moreover, note that we can also write all the other U_g in this form. Notice that for any k, $\deg V_k = n := |G/H|$. We proceed with a theorem similar to 3.3.

Theorem 3.12. For each k, $V_k(Y)$ is a polynomial in $K[Y]$.

Proof. Let $g' \in G/H$

$$V_j(Y)^{g'} = \prod_{g\in G/H} (Y - (\vartheta_j^g)^{g'}) = \prod_{g\in G/H} (Y - \vartheta_j^{g'g}) = \prod_{\tilde{g}\in g'G/H} (Y - \vartheta_j^{\tilde{g}}) = V_j(Y)$$

where the penultimate equality is because $G/H = g'G/H$. □

Now, the V_k are not necessarily irreducible. Moreover, since H is normal, the extension L/K is Galois with Galois group G/H. Hence, $[L : K] = |G/H| =: n$. We make the following observation:

Theorem 3.13. Fix k. Let $H' \geq H$ be the largest subgroup of G which fixes the ϑ_k. Then V_k decomposes into $[H' : H]^{\text{th}}$ powers of linear factors as a polynomial in L and moreover, G/H' fixes V_k.

Proof. Let $\mathcal{K}$ be the intermediate field corresponding to H', that is $H' = \text{Gal}(M/\mathcal{K})$. Any element $g \in G/H$ can be written as $g = hg'$ where $h \in H'/H$ and $g' \in G/H'$. And so, we rewrite

$$V_k(Y) = \prod_{g\in G/H} (Y - \vartheta_k^g) = \prod_{g'\in G/H'} \prod_{h\in H'/H} (Y - \vartheta_k^{hg'}) = \prod_{g'\in G/H'} (Y - \vartheta_k^{g'})^{[H':H]}$$

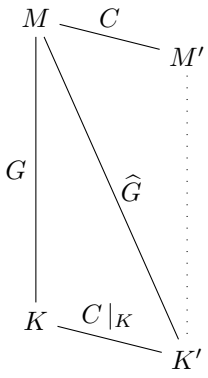
where the last inequality is due to the fact that H' fixes ϑ_k. Moreover, G/H' fixes V_k. □

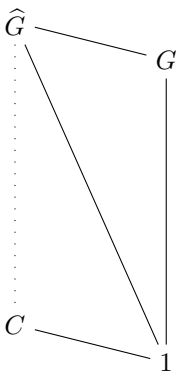
Corollary 3.14. If $\gcd(V_k, V_k') = 1$, then V_k is a defining polynomial for L/K.

Proof. From theorem 3.13, if $\gcd(V_k, V_k') = 1$, then $[H' : H] = 1$ and so $H = H'$ is the largest subgroup of G fixing ϑ_k. Notice that since from theorem 3.12 $V_k(Y) \in K[Y]$. Moreover, $V_k(Y)$ is an $n = |G/H|$ degree polynomial fixed by G/H, and $\vartheta_k \in L$ since $W \in L[X]$ by theorem 3.3. Thus, it is a defining polynomial for L/K. □

3.1.2 Non-Galois Case

In this section, we show that the techniques of the previous section works in some field extensions which might not necessarily be Galois.





Let M'/K', a finite separable field extension. Take M to be smallest field such that M/K' is Galois which contains M'. This is called the Galois closure of M' over K'. Since M' satisfies $K' \leq M' \leq M$, by fundamental theorem of Galois theory, we obtain a subgroup C of $\widehat{G}$ which fixes M'. That is,

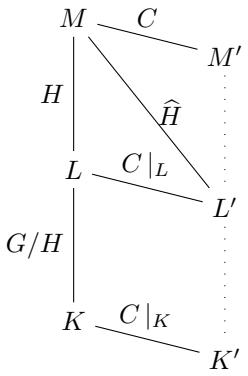
$$C = \text{Gal}(M/M').$$

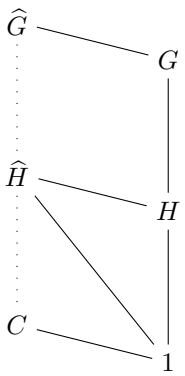
We make the following assumption.

Assumption 3.15. Assume there exists a normal subgroup G of $\widehat{G}$ such that

$$\widehat{G} = G \rtimes C.$$

In other words, G is a normal complement of C. As G is a subgroup of $\widehat{G}$, it corresponds to a field K such that $G = \mathrm{Gal}(M/K)$. As G is normal, then the extension K/K' is also Galois with Galois group $\widehat{G}/G$, which is isomorphic to $C|_K$. Now, take H to be a normal subgroup of G. Then it corresponds to an intermediate field L.





Then $H = \mathrm{Gal}(M/L)$ and L/K is Galois with group G/H because H is normal. We make the following second assumption.

Assumption 3.16. Assume that $chc^{-1} \in H$ for any $h \in H$ and $c \in C$, that is C normalizes H.

With this assumption, any element $ch \in \langle H, C\rangle$ can be rewritten as

$$ch = cc^{-1}h'c = h'c$$

where $h' \in H$. And hence

$$\langle H, C\rangle = H \rtimes C.$$

Now, take L' to be the fixed field of the subgroup $\widehat{H} = \langle H, C\rangle = H \rtimes C$ of $\widehat{G}$. Since $C \leq \widehat{H} \leq \widehat{G}$, then $K' \leq L' \leq M'$. Moreover, take note that since H is a normal subgroup of $\widehat{H}$, L/L' is Galois of group $\mathrm{Gal}(L/L')$. Now, note that

$$[\widehat{G} : \widehat{H}] = [G : H] \qquad \text{and} \qquad [\widehat{H} : C] = [H : 1]\,. \tag{3.17}$$

This means that

$$[L' : K'] = [L : K] \qquad \text{and} \qquad [M' : L'] = [M : L]. \tag{3.18}$$

In particular,

$$[M' : K'] = [M : K]. \tag{3.19}$$

As we see with the two theorems, the field extensions $M/L/K$ and $M'/L'/K'$ have the same defining polynomials if $x \in M'$.

Theorem 3.20. Let $f \in K'[X]$ be an defining polynomial for M'/K'. That is, it has a root $x \in M'$ such that $M' = K'(x)$. Then f is also a defining polynomial for the extension M/K.

Proof. Note that

$$M = K \cdot M' = K \cdot K'(x) = K(x).$$

And by equation (3.19), $[M : K] = \deg f$. This means that f is still irreducible in K and thus is also a defining polynomial for M/K. □

Theorem 3.21. Let $f \in K[X]$ be an defining polynomial for M/K with a root $x \in M'$. That is, $M = K(x)$. Then f is also a defining polynomial for the extension M/K.

Proof. If x is the root in M', we can write

$$f(X) = \prod_{g \in G} (X - x^g).$$

Moreover, for any $c \in C$,

$$f(X)^c = \prod_{g \in G} (X - (x^g)^c) = \prod_{g \in G} (X - x^{cg}) = \prod_{g \in G} (X - x^{cgc^{-1}c}) = \prod_{g' \in cGc^{-1}} (X - x^{g'c}) = \prod_{g' \in cGc^{-1}} (X - x^{g'}) = f(X)$$

where the penultimate equality is due to the fact that $x \in M'$ and hence is fixed by c, and the last equality is due to the fact that C normalizes G. That is, $G = cGc^{-1}$. Again, by equation (3.19), we find that f defines M'/K'. □

In fact, if we find an appropriate C and the appropriate M'/K', we can do most of the computations we did in subsection 3.1.1 within the smaller fields M'/K'. The goal of the remainder of this section is to show tht we can indeed do this. Let U_{gH} be defined as in (3.9). Then, the action of $c \in C$ on U_{gH} is defined as follows:

$$U_{gH}^c = \prod_{h \in H} (X - (x^{hg})^c) = \prod_{h \in H} (X - x^{chg}) = \prod_{h \in H} (X - x^{ch(c^{-1}c)g(c^{-1}c)}) = \prod_{h \in H} (X - x^{chc^{-1}cgc^{-1}}).$$

Since C normalizes H, we have

$$\prod_{h \in H} (X - x^{chc^{-1}cgc^{-1}}) = \prod_{h \in cHc^{-1}} (X - x^{h'cgc^{-1}}) = U_{gH}.$$

In particular $W = U_{1_G}$ is stable in C. And hence $W \in L'[X]$. Similar to the Galois case, since W is a polynomial in $L'[X]$ of degree $|H| = [M' : L']$ with a root $x \in M'$ such that $M' = L'(x)$. Hence, W is and generates M'/L'. Now, for V_k, notice that the action of C on the ϑ_k^g is the same as the action of C on the U_{gH}. So G and C permute the ϑ_k^g and thus $V_k \in K'[Y]$. If V_k is irreducible, then it generates L'/K'. Thus, we have done exactly what we did in subsection 3.1.1 but for the non-Galois extension $M'/L'/K'$. And we have found $W \in L'[X]$ and $V \in K'[Y]$ which define the extensions M'/L' and L'/K', respectively.

3.2 Hecke Representation

If $K' = \mathbb{Q}$ and $M' = \mathbb{Q}(g(\sqrt{-D}))$, then we can just compute V and end up with a polynomial in $\mathbb{Q}[X]$. Moreover, it can be shown that $g(\sqrt{-D})$ is an integral element. Hence, V is in fact a polynomial in $\mathbb{Z}[X]$ since all Galois conjugates appearing in the coefficients of V are also integral. Using the appropriate precision, we should be able to compute V such that the coefficients will be exactly or almost integers.
In contrast, however, $U \in L'[X]$ is not as easy to represent. Unlike V, it does not lie in $\mathbb{Q}[X]$, but in $\mathbb{R}[X]$. In practice, all of our computations involve approximations in the form of floating point numbers. Most of the time, however, we want to work in terms of the generating element α of the field extension. Hence, we want to express the coefficients in terms of the element ϑ. Thankfully, we have a lemma that solves this problem.

Lemma 3.22. Let L/K be a separable field extension of degree n with primitive element α. Denote by (α_i) the conjugates of α, and write the defining polynomial of the extension L/K as

$$V(Y) = \prod_{i=0}^{n-1} (Y - \alpha_i).$$

Then, we can write any $\vartheta \in L$ as

$$\vartheta = \frac{g_\vartheta(\alpha)}{V'(\alpha)}$$

for some $g_\vartheta \in K[Y]$. Moreover, if $\mathcal{O}_K$ is an integrally closed subring of K and α and ϑ are integral over $\mathcal{O}_K$, then $V, g_\vartheta \in \mathcal{O}_K[Y]$.

Proof. If we write $(\vartheta_i)_{0 \le i < n}$ be the conjugates of ϑ, we can let

$$g_\vartheta = \sum_{i=0}^{n-1} \vartheta_i \frac{V(Y)}{Y - \alpha_i}. \tag{3.23}$$

Note that applying any automorphism $\sigma \in \mathrm{Gal}(L/K)$ to g_ϑ will only permute the terms of the summation. Hence, $g_\vartheta \in K[Y]$. Moreover,

$$g_\vartheta(\alpha) = \vartheta \cdot V'(\alpha). \tag{3.24}$$

If $\alpha \in \mathcal{O}_K$, then so will the conjugates of α. In particular,

$$\frac{V(Y)}{(Y - \alpha_i)} = \tilde{f}_i(Y) = \frac{V(Y)}{Y - \alpha_i} = \prod_{\substack{j=0,\dots,n-1 \\ j \neq i}} (Y - \alpha_i) \in \mathcal{O}_K[Y]$$

for any $i = 0, 1, \dots, n-1$. Assuming further that $\vartheta \in \mathcal{O}_K$, then so will the conjugates ϑ_i. From this, we conclude that $g_\vartheta \in \mathcal{O}_K[Y]$.

□

However, we can use a divide and conquer approach to compute g_ϑ. We construct a tree of height $k = \lceil \log_2 n \rceil$ whose leaves are $Y - \alpha_i$. If n is not a power of 2, we add the leaves of value 1 so that we have $n' = 2^k$ leaves. From here, we compute the value for the other nodes by taking the product of their children. Now, we discuss how to compute V and all its subproducts. This is taken care of by the following algorithm, taken from [9].

Algorithm 3.25 (Subproduct tree)**.** `subproductTree`

INPUT: An integer k and a list of polynomials $f_0, \dots, f_{n-1}$. Assume n is a power of 2. Otherwise, add 1's to the end of the list.

OUTPUT: A binary tree T, represented in an array of floating point numbers, whose leaves are the input and each node T' of T is the product of all the leaves in the subtree with root T'. In particular, the root T is the product of all the leaves.

ALGORITHM

1. Initialize a 1-dimensional array $T = [T_1, \dots, T_{2n-1}]$. Set $T_{n+k} = f_k$ for $k = 0, \dots, n-1$.
2. Set T_i with $1 \le i < n$ to be $T_{2i} T_{2i+1}$.
3. Return T.

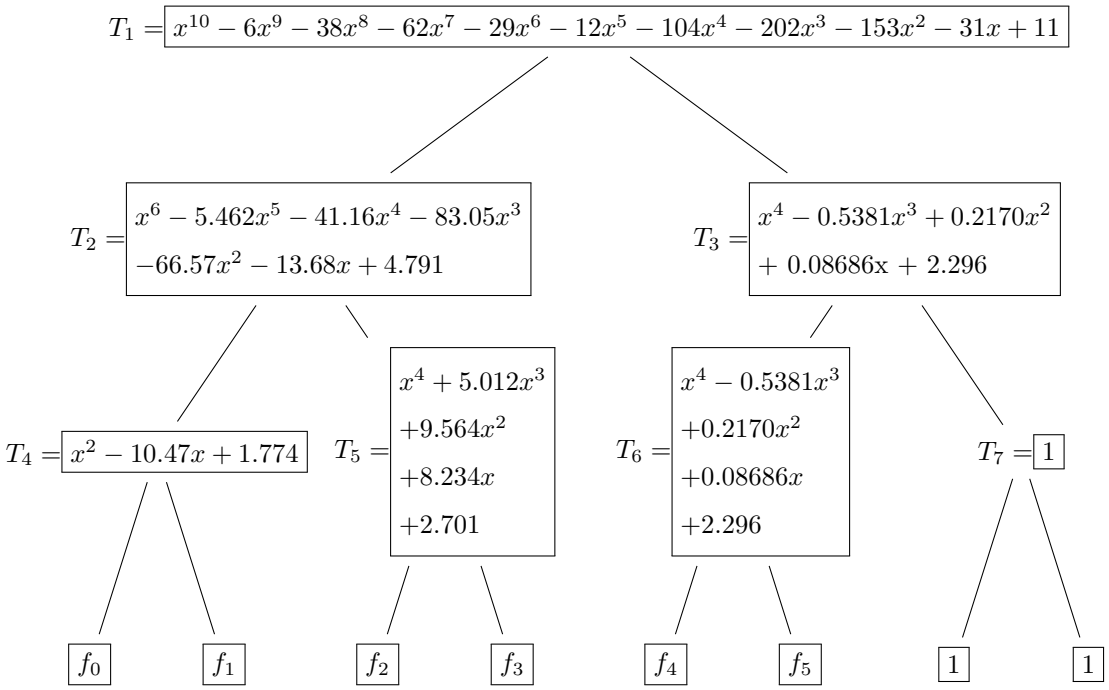
Remark 3.26. The leaves of the subtree T' whose root is T_s are exactly $T_{js+\ell}$ where

- $t = k - \lfloor \log_2 s \rfloor$ is the height of the tree T', and
- $j = 2^t$ is the number of leaves of T'.

Example 3.27. If we let $f_i = Y - \alpha_i$ be the leaves of T, we obtain a tree such that its root is $T_1 = V(Y)$.

Example 3.28. Below is a concrete example of a subproduct tree for $f_0, f_1, f_2, f_3, f_4, f_5$ where

$$f_0 = x - 10.30$$
$$f_1 = x - 0.1722$$
$$f_2 = x^2 + 2.437x + 1.587$$
$$f_3 = x^2 + 2.575x + 1.702$$
$$f_4 = x^2 + 1.443x + 1.272$$
$$f_5 = x^2 - 1.982x + 1.806$$



Now that we have precomputed the subproducts, we utilise the tree we made using another algorithm in [9].

Algorithm 3.29 (Linear combination for linear moduli)**.** `linearCombinationForLinearModuli`

INPUT:

- A list $c = [c_0, \ldots, c_{n-1}]$.
- T, a tree.
- An integer s.

OUTPUT:

$$\sum_{\ell=0}^{j-1} c_{(j(s-1)+\ell} \frac{T_s}{f_{j(s-1)+\ell}}$$

where j is as in remark 3.26.

ALGORITHM

1. If $s > 2^k + (n-1)$ (i.e. $T_s = 1$), return 0.
   If $s \geq 2^k$, return c_{s-2^k}.
   Otherwise, proceed with the algorithm.
2. Let f be the result of `linearCombinationForLinearModuli` with $c, k, T, 2s$.

3. If $T_{2s+1} = 1$, return f.

4. Let g be the result of `linearCombinationForLinearModuli` with $c, k, T, 2s+1$.

5. Return $f \cdot T_{2s+1} + g \cdot T_{2s}$.

Example 3.30. Let $c_i = \vartheta_i$ and T be as in example 3.27. Feeding these parameters and $s = 1$ to algorithm 3.29, we should obtain g_ϑ, as in (3.23). For reference, we write it as the following algorithm 3.31.

Algorithm 3.31 (Computing for g_ϑ)**.** `gv`

INPUT:

- A list $a = (\alpha_0, \ldots, \alpha_{n-1})$ of conjugates of the primitive element $\alpha := \alpha_0$ of L.
- A list $c = (\vartheta_0, \ldots, \vartheta_{n-1})$ of conjugates of $\vartheta := \vartheta_0$ in L.

OUTPUT: $g_\vartheta(Y)$, a polynomial such that $\vartheta = g_\vartheta(\alpha)/V'(\alpha)$.

ALGORITHM

1. Let T be the subproduct tree obtained by `buildSubproductTree` using $Y - \alpha_i$ as the leaves.

2. Return the output of `linearCombinationForLinearModuli` with parameters a, c, $k = \lceil \log_2 n \rceil$, T and $s = 0$.

Now, we discuss an example of algorithm 3.29 which will be vital in computing g_ϑ in a quicker way.

Example 3.32. Let $\alpha \in \mathbb{C}$. We build the subproduct tree with leaves $f_0 = Y - \alpha$ and $f_1 = Y - \overline{\alpha}$. Thus, for $i = 0, 1$, $T_{2+i} = f_i$. From this, we conclude that the root of the tree T is

$$T_1 = (Y - \alpha)(Y - \overline{\alpha}) = Y^2 - \operatorname{tr}(\alpha)Y + N(\alpha) \in \mathbb{R}[Y].$$

Moreover, if we use this tree, let $(c_0, c_1) = (\vartheta, \overline{\vartheta})$ and $s = 1$, we obtain as output from algorithm 3.29:

$$\sum_{\ell=0}^{2-1} c_{2(1-1)+\ell} \frac{T_1}{f_{2(1-1)+\ell}} = \vartheta \cdot \frac{T_1}{Y - \alpha} + \overline{\vartheta} \cdot \frac{T_1}{Y - \overline{\alpha}} = \vartheta(Y - \overline{\alpha}) + \overline{\vartheta}(Y - \alpha). \tag{3.33}$$

Note that equation (3.33) is in $\mathbb{R}[Y]$ since we have

$$\vartheta(Y - \overline{\alpha}) + \overline{\vartheta}(Y - \alpha) = (\vartheta + \overline{\vartheta})Y - (\vartheta\overline{\alpha} + \overline{v}\alpha).$$

And indeed,

$$\begin{aligned} \vartheta + \overline{\vartheta} &= 2 \cdot \operatorname{Re}(\vartheta) \\ \vartheta\overline{\alpha} + \overline{\vartheta}\alpha &= 2 \operatorname{Re}(\vartheta) \operatorname{Re}(\alpha) + 2 \operatorname{Im}(\vartheta) \operatorname{Im}(\alpha) \end{aligned}$$

are both real numbers.

In our motivating example, C is the group generated by conjugation. By virtue of example 3.32, we will be able to limit all computations to only involve real numbers. This can be achieved by merging all leaves corresponding to pairs of conjugate roots as

$$\tilde{f} = Y^2 - \operatorname{tr}(\alpha)Y + N(\alpha)$$

and their corresponding c_i's be merged into

$$\tilde{c} = 2(\operatorname{Re}(\vartheta))Y - 2(\operatorname{Re}(\vartheta) \operatorname{Re}(\alpha) + \operatorname{Im}(\vartheta) \operatorname{Im}(\alpha)). \tag{3.34}$$

Using this modification, all our computations will exclusively involve real numbers.

Chapter 4

The Algorithm

Recalling what we have from the previous section, M'/K' is a finite separable field extension generated by the polynomial f. K' and M' have corresponding fields K and $M = M'K$ where the extension M/K is Galois with Galois group $G = \mathrm{Gal}(M/K)$. Taking a normal subgroup H of G enables us to compute the polynomials U_{gH} as in (3.9). In particular, we let $U := U_{\overline{1}}$. In our motivating example where $K' = \mathbb{Q}$, $K = \mathbb{Q}(\sqrt{-D})$, $M = H_K$, we expect to represent the coefficients of U as floating point numbers. Next, we can compute $V \in \mathcal{O}_K[X] = \mathbb{Z}[X]$ as in (3.11), rounding to integers in order to lessen the effect of rounding errors. Finally, we wish to express the coefficients of U in terms of a primitive element of L'. We achieve this representation by applying 3.31 to each of the coefficients of U. Let W be the polynomial represented in this way. At this point, we would have obtained polynomials $V \in K'[Y]$ and $W \in L'[X]$. These are minimal polynomials for the extensions L/K and M/L, respectively. And from our previous discussion, they are also the polynomials for the extensions L'/K' and M'/L'. Moreover, $G/H \cong \mathrm{Gal}(L/K)$. Hence, if we have a normal subgroup $\tilde{H}/H$ of G/H, then we can again find another intermediate field $\tilde{L}'$ whose corresponding $\tilde{L}$ has Galois group $\frac{G/H}{\tilde{H}/H} \cong G/\tilde{H}$. We can repeat this until we eventually run out of normal subgroups (G is a finite group, after all). This algorithm is written more formally as 4.19, which includes all the improvements we introduce in this chapter. Let us first decide the normal subgroups to be used in each iteration. We define a normal series.

Definition 4.1. A normal series of a group G is a sequence of normal subgroups of G such that

$$H_0 := 1 \lhd H_1 \lhd H_2 \lhd \cdots \lhd H_t := G. \tag{4.2}$$

Moreover, each H_i/H_{i-1} is called a factor group.

The longer the normal series we take, the more iterations we have. This will then lead to intermediate fields which have small indices, which is advantageous. And so, we define a maximal normal series as follows:

Definition 4.3. A composition series of a group G is a normal series (as in (4.2)) of finite length such that its factor groups are all simple (i.e. their only proper normal subgroup is the trivial group).

Remark 4.4. A normal series is a composition series if and only if it is of maximal length.

Example 4.5. If G is a finite abelian group, then the factor groups of any composition series of G are cyclic groups of prime order.

Example 4.6 (Illustration)**.** Let

$$H_0 := 1 \lhd H_1 \lhd H_2 \lhd \cdots \lhd H_t := G. \tag{4.7}$$

be a composition series for the finite abelian group $G = \mathrm{Gal}(M/K)$. For the ι^{th} iteration, our input will involve the groups

$$G^{(\iota)} = \frac{G}{H_{\iota-1}} \qquad \text{and} \qquad H^{(\iota)} = \frac{H_\iota}{H_{\iota-1}}.$$

Since (4.7) is a composition series, all the H_i's are normal subgroups of G and hence $H^{(\iota)} \lhd G^{(\iota)}$.

We illustrate the progression of the diagram after each iteration of our algorithm. Here $G^{(\iota)}$ and $H^{(\iota)}$ are the Galois group and the normal subgroup, $\mathcal{L}^{(\iota)}$ is a list of the elements of $G(\iota)$, expressed symbolically, and $\mathcal{R}^{(\iota)} = \mathcal{R}_P$ is a list of roots of P.

Original Situation

Iteration #1

INPUT:

- $G^{(1)} = G$
- $H^{(1)} = H_1$
- $\mathcal{L}^{(1)}$
- $\mathcal{R}^{(1)} = \mathcal{R}_V$

OUTPUT:

- $G^{(2)} = G/H_1$
- $\mathcal{L}^{(2)}$
- $\mathcal{R}^{(2)} = \mathcal{R}_{V^{(2)}}$
- W_1

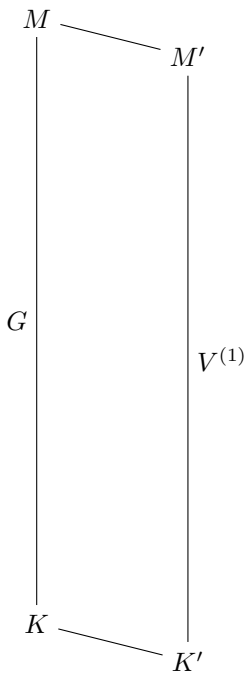
Iteration #2

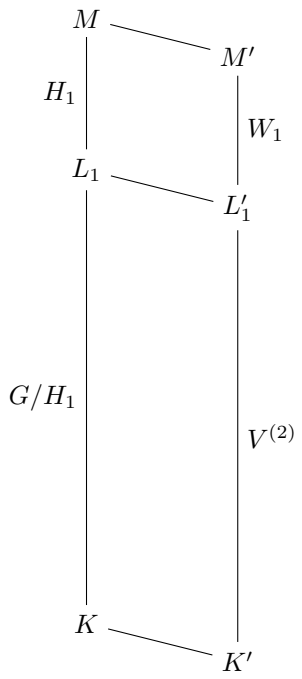
INPUT:

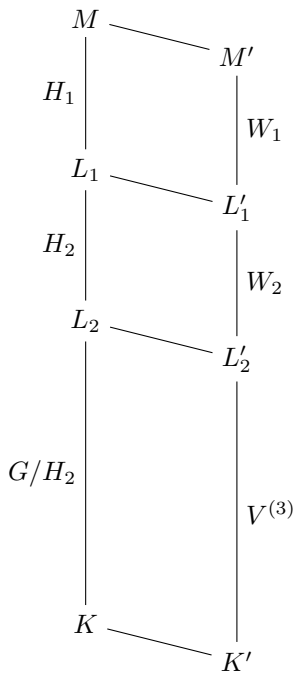
- $G^{(2)} = G/H_1$
- $H^{(2)} = H_2$
- $\mathcal{L}^{(2)}$
- $\mathcal{R}^{(2)} = \mathcal{R}_V$

OUTPUT:

- $G^{(3)} = G/H_2$
- $\mathcal{L}^{(3)}$
- $\mathcal{R}^{(3)} = \mathcal{R}_V$
- W_2







4.1 Ordering the Roots

In practice, we do not even need to compute the input polynomial V. We only use an array of roots of$V$. We want a way to organize the list so that we know which values to use in computing the U_{gH}'s defined in (3.9). In particular, we want a way to order the list of roots such that

$$\mathcal{L} = (x_0, x_1, \ldots, x_{mn-1}).$$

so that we can rewrite the set of equations (3.9) as

$$U_i = \prod_{j=0}^{m-1} (X - x_{im+j}) = \sum_{j=0}^{|H|-1} (-1)^{m-j} \vartheta_{i,j} X^j \qquad \text{for all } i = 0, 1, \ldots, n-1. \tag{4.8}$$

Moreover, we want to distinguish $W \in L'[X]$ and so we impose that $(x_0, \ldots, x_{m-1})$ be exactly the roots of U (i.e. the x^h's for $h \in H$) so that we can rewrite (3.2) as

$$W = U_0 = \prod_{j=0}^{m-1} (X - x_j).$$

Consequently, the equation (3.11) for V can now be written as

$$V = V_k = \prod_{i=0}^{n-1} (Y - \vartheta_{i,k})$$

where k is chosen in the algorithm. It turns out that not only is it possible to obtain an ordering that works for the first iteration, but we can actually find a way to reorder these roots in such a way that the first m components are always those associated to the subgroup H of that iteration.

4.1.1 Cyclic Case

Suppose first that G is a cyclic group. We can write it as

$$G = \langle g \rangle.$$

Taking a composition series of G, we obtain

$$H_0 := 1 \lhd H_1 \lhd H_2 \lhd \cdots \lhd H_t := G = \langle g \rangle. \tag{4.9}$$

This means that for each $\iota = 1, \ldots, t$, we have

$$|H_\iota / H_{\iota-1}| = p_\iota$$

for some prime p_ι. Moreover, we can write

$$H_\iota = \langle g^{|G|/p_1 \cdots p_\iota} \rangle$$

For the ι^{th} iteration. We must be able to partition the vector into blocks of length $|H_\iota|$ such that each block contains elements of a coset in G/H. We obtain this vector as follows.

Algorithm 4.10 (Ordered List)**.** `orderedList`

INPUT: A cyclic group $G = \langle g \rangle$ and a composition series for G as in (4.2).

OUTPUT: An ordered list $\mathcal{L}$ of elements of G.

ALGORITHM:

1. Initialize $\mathcal{L} = (e)$.
2. For each $\iota = 1, \ldots, t$:
   (a). Let $\{a_0, \ldots, a_{p_\iota - 1}\}$ be a list of coset representatives of $H_\iota / H_{\iota-1}$ with $a_0 = e$, the representative of the trivial element.
   (b). Let $\tilde{\mathcal{L}}$ be the concatenation of $a_0\mathcal{L} = \mathcal{L}, a_1\mathcal{L} \ldots, a_{p_\iota - 1}\mathcal{L}$. Replace $\mathcal{L}$ by $\tilde{\mathcal{L}}$.
3. Output $\mathcal{L}$.

At the start of iteration ι, it is ensured that the elements of the list $\mathcal{L}$ are precisely the elements of $H_{\iota-1}$. And since the old $\mathcal{L}$ is the prefix of the new $\mathcal{L}$, it is ensured that the first $|H_{\iota-1}|$ elements of $\mathcal{L}$ are still the elements of $H_{\iota-1}$. Moreover, partitioning $\mathcal{L}$ into subvectors of length $|H_\iota|$, each block should correspond to the cosets in G/H_ι.

Example 4.11. Suppose $G = \langle b \rangle$ of order 10. A composition series for G is

$$H_0 := \langle e \rangle \quad \lhd \quad H_1 := \langle b^5 \rangle \quad \lhd \quad H_2 := \langle b \rangle$$

We start with $\mathcal{L} = (e) = (b^0)$. Note that $\{b^0, b^5\}$ is a set of representatives for H_1/H_0. And so we have the following after the first iteration

$$\mathcal{L} = (b^0 \cdot b^0, b^0 \cdot b^5) = (b^0, b^5).$$

Finally, $\{b^0, b^1, b^2, b^3, b^4\}$ is a set of representatives for H_2/H_1. And so we have

$$\mathcal{L} = \left(b^0\mathcal{L}, b^1\mathcal{L}, b^2\mathcal{L}, b^3\mathcal{L}, b^4\mathcal{L}\right) = (b^0, b^5, b^1, b^6, b^2, b^7, b^3, b^8, b^4, b^9).$$

Note that

$$\underbrace{(\underbrace{b^0}_{\in H_0}}_{\in H_1}, b^5, b^1, b^6, b^2, b^7, b^3, b^8, b^4, b^9)$$

and partitioning into blocks of length $|H_1| = 2$, we obtain elements of each coset in G/H_1.

4.1.2 General Case

Now, for the general case of a finite abelian group G, writing

$$G = C_1 \times \ldots \times C_r,$$

we can just "concatenate" the composition series for each cyclic group to obtain a composition series for G. For example, if $r = 2$, we can have

$$H_{1,0} \lhd H_{1,1} \lhd \cdots \lhd H_{1,t_1} = C_1 \lhd C_1 \times H_{2,1} \lhd \cdots \lhd C_1 \times H_{2,t_2} = C_1 \times C_2 = G. \tag{4.12}$$

Hence, we can modify 4.10 by taking instead a group G with a composition series such as (4.12).

Example 4.13. Let $G = C_1 \times C_2 = \langle a \rangle \times \langle b \rangle$ where a and b are of order 2 and 10, respectively. And so we have a composition series

$$H_0 = \langle e \rangle \quad \lhd \quad H_1 = \langle a \rangle \quad \lhd \quad H_2 = \langle a \rangle \times \langle b^5 \rangle \quad \lhd \quad H_3 = \langle a \rangle \times \langle b \rangle. \tag{4.14}$$

We start again with $\mathcal{L} = (e)$. And then take $\{a^0, a^1\}$ to be the coset representatives of H_1/H_0. And so we obtain

$$\mathcal{L} = (a^0, a^1).$$

That takes care of C_1. In a manner similar to what happened in example 4.11, we obtain

$$\mathcal{L} = (\underbrace{\underbrace{\underbrace{1}_{\in H_0}}_{\in H_1}}_{H_2}, a, b^5, b^5a, b, ab, b^6, ab^6, b^2, ab^2, b^7, ab^7, b^3, ab^3, b^8, ab^8, b^4, ab^4, b^9, ab^9). \tag{4.15}$$

And so, at this point, we would be able to find an appropriate order of the roots to be used for the main algorithm.

Example 4.16. The ideal class group of $K = \mathbb{Q}(\sqrt{-D})$ when $D = 455$ is isomorphic to

$$\mathrm{Cl}(-4D) = C_2 \times C_{10} = \langle \mathbf{a} \rangle \times \langle \mathbf{b} \rangle. \tag{4.17}$$

where $\mathbf{a} = [5, 0, 91]$ and $\mathbf{b} = [3, 2, 152]$. This is because in the special case of $D \equiv 7 \pmod 8$, the ideal class group $\mathrm{Cl}(-D)$ is isomorphic to $\mathrm{Cl}(-4D)$, as stated in 2.13. Using our computations in 2.14, we are then able to obtain the following order for the roots of $u(\sqrt{-D})$.

i	Q		x_i		i	Q		x_i
0	$[1, 0, 455]$	1	11.54		10	$[1, 0, 455]$	$\mathbf{b}^7$	$-0.2266 - 0.8176i$
1	$[5, 0, 91]$	$\mathbf{a}$	-1.236		11	$[5, 0, 91]$	$\mathbf{ab}^7$	$-0.4952 - 0.04877i$
2	$[13, 0, 35]$	$\mathbf{b}^5$	-0.8816		12	$[13, 0, 35]$	$\mathbf{b}^3$	$-0.2266 + 0.8176i$
3	$[7, 0, 65]$	$\mathbf{ab}^5$	1.054		13	$[7, 0, 65]$	$\mathbf{ab}^3$	$-0.4952 + 0.04877i$
4	$[3, 2, 152]$	$\mathbf{b}$	$-1.374 - 1.153i$		14	$[3, 2, 152]$	$\mathbf{b}^8$	$-0.6623 - 0.7015i$
5	$[15, -10, 32]$	$\mathbf{ab}$	$0.1554 + 0.8326i$		15	$[15, -10, 32]$	$\mathbf{ab}^8$	$-0.6251 + 0.4916i$
6	$[16, 6, 29]$	$\mathbf{b}^6$	$0.2375 - 0.6643i$		16	$[16, 6, 29]$	$\mathbf{b}^4$	$0.2375 + 0.6643i$
7	$[21, 14, 24]$	$\mathbf{ab}^6$	$0.7532 - 0.2435i$		17	$[21, 14, 24]$	$\mathbf{ab}^4$	$0.7532 + 0.2435i$
8	$[9, -4, 51]$	$\mathbf{b}^2$	$-0.6623 + 0.7015i$		18	$[9, -4, 51]$	$\mathbf{b}^9$	$-1.374 + 1.153i$
9	$[19, 2, 24]$	$\mathbf{ab}^2$	$-0.6251 - 0.4916i$		19	$[19, 2, 24]$	$\mathbf{ab}^9$	$0.1554 - 0.8326i$

4.2 Exploiting Complex Conjugation

By inspecting example 4.16, one can observe that the conjugate roots come in pairs. As we can see from the previous example, we could have gotten away with computing less because some of the x_i's are conjugates of each other. This is due to the fact that the action of conjugation (2.17). To be able to keep track of which x_i's are conjugates of each other, we need to look at the corresponding group element which corresponds to x_i. This is a symbolic representation of the group structure embedded in the x_i's. For every pair of elements which are inverses of each other, we need only to compute the corresponding x_i for one of them! This is even more important whilst computing the U_i's. Recall that the U_i's each correspond to an element of G/H. Just like in the case of the x_i's, we can only compute one U_i in each pair which correspond to inverse elements. We recover the rest by taking conjugates.

Example 4.18. Continuing from the example 4.16, we see that we need not compute the other U_i's. Here is a table with $H = H_1$. It lists down the coset of G/H alongside the corresponding U_{gH}, which in the algorithm we will refer to as U_i.

i	elements of coset		U_i
0	$\{e, \mathbf{a}\}$	$(X - x_0)(X - x_1)$	$X^2 - 10.302X - 14.260$
1	$\{\mathbf{b}^5, \mathbf{ab}^5\}$	$(X - x_2)(X - x_3)$	$X^2 - 0.17219X - 0.92899$
2	$\{\mathbf{b}, \mathbf{ab}\}$	$(X - x_4)(X - x_5)$	$X^2 + (1.2185 + 0.32027i)X + (0.74629 - 1.3230i)$
3	$\{\mathbf{b}^6, \mathbf{ab}^6\}$	$(X - x_6)(X - x_7)$	$\overline{U_8}$
4	$\{\mathbf{b}^2, \mathbf{ab}^2\}$	$(X - x_8)(X - x_9)$	$X^2 + (1.2875 - 0.20988i)X + (0.75890 - 0.11293i)$
5	$\{\mathbf{b}^7, \mathbf{ab}^7\}$	$(X - x_{10})(X - x_{11})$	$\overline{U_6}$
6	$\{\mathbf{b}^3, \mathbf{ab}^3\}$	$(X - x_{12})(X - x_{13})$	$X^2 + (0.72171 - 0.86640i)X + (0.072306 - 0.41591i)$
7	$\{\mathbf{b}^8, \mathbf{ab}^8\}$	$(X - x_{14})(X - x_{15})$	$\overline{U_4}$
8	$\{\mathbf{b}^4, \mathbf{ab}^4\}$	$(X - x_{16})(X - x_{17})$	$X^2 + (-0.99075 - 0.90779i)X + (0.017132 + 0.55818i)$
9	$\{\mathbf{b}^9, \mathbf{ab}^9\}$	$(X - x_{18})(X - x_{19})$	$\overline{U_2}$

Moreover, we can use the trick inspired by example 3.32 to also speed up the computation of the g_ϑ's.

4.3 The Main Algorithm

After all the discussion, we are now ready to state the main algorithm.

Algorithm 4.19. `mainAlgorithm`

Input:

- An extension of fields M'/K' generated by the polynomial V with $d := \deg V$.
- An ordered list $\mathcal{L}$ of the elements of the group $G = \mathrm{Gal}(M/K)$, ordered using algorithm 4.10 using the composition series
$$\langle e \rangle = H_0 \lhd H_1 \lhd \cdots \lhd H_{t-1} \lhd H_t = G.$$
of G. Note that $|G| = d$.
- An ordered list $\mathcal{R}_V = (x_0, \ldots, x_{d-1})$ of the roots of V obtained from taking the corresponding roots of $\mathcal{L}$.

Output:

- A list of intermediate fields L'_{i-1}/L'_i for $i = 1, \ldots, t$ (where $M_0 = L'_0$ and $K_0 = L'_t$) and their respective irreducible polynomials $W_i \in L_i[X]$.

Algorithm

Let $G^{(0)} = G$, $H^{(0)} = H_0$, $\mathcal{L}^{(0)} = \mathcal{L}$. For each $\iota = 1, 2, \ldots, t-1$, do the following:

1. Set
$$G^{(\iota)} = \frac{G^{(\iota-1)}}{H^{(\iota-1)}} \cong \frac{G}{H_{\iota-1}} \qquad \text{and} \qquad H^{(\iota)} = \frac{H_\iota}{H_{\iota-1}}$$
and let
$$m = |H^{(\iota)}| \qquad \text{and} \qquad n = |G^{(\iota)}|/m.$$
2. Determine $\mathcal{I} \subseteq \{0, \ldots, n-1\}$, the set of indices for which we must compute U_i.
   (a). Set $\mathcal{L}^{(\iota)}$ be a list of coset representatives of $G^{(\iota)}/H^{(\iota)}$.
   This can be obtained by taking every m^{th} element of $\mathcal{L}^{(\iota-1)}$, starting from the first.

(b). For each pair of inverse elements in $G^{(\iota)}/H^{(\iota)}$., choose one and put its index in $\mathcal{I}$. Moreover, remember which indices are paired with each other. [1]

3. Compute the U_i's for this iteration.

   (a). For each $i \in \mathcal{I}$, compute U_i's as in (4.8) using $\mathcal{R}_V$.

   (b). For each $i \notin \mathcal{I}$, compute U_i by taking the conjugate of the appropriate polynomial. This polynomial would have been computed in the previous step.

4. Find $V^{(\iota+1)}$.
   For each $j = m-1, \ldots, 0$:

   (a). Construct the subproduct tree whose $|\mathcal{I}|$ leaves are the $\vartheta_{i,j}$'s following the notation in (4.8) using the real floating point arithmetic as described in example 3.32.

   (b). Check if $V^{(\iota+1)} = T_1$ is irreducible. If it is, set $k := j$ and of course, set $V^{(\iota+1)} = T_1$.

5. Compute
$$W_\iota = X^m + \sum_{i=0}^{m-1} (-1)^{m-i} \frac{g_i(Y)}{(V^{(\iota+1)})'(Y)} X^i$$
   by using algorithm 3.31 to express the coefficients of U_0 as field elements.

6. If $\iota = t-1$, let $W_t = V^{(t)}$. Otherwise, let $\mathcal{R}_V$ be the roots of $V^{\iota+1}$

Example 4.20. Let $K = \mathbb{Q}(\sqrt{-D})$ where $D = -455$. Let $M = H_K$, the Hilbert class field of K. Moreover, we write
$$G^{(1)} = G \qquad \text{and} \qquad H^{(1)} = H_1$$
From our discussion in section 3.1, we have the corresponding $M' = \mathbb{Q}(j(\mathfrak{k})) = \mathbb{Q}(u(Q))$ and $K' = \mathbb{Q}$ where $u = f/\sqrt{2}$. Note that we choose $\mathfrak{k}$ to be the trivial ideal class (i.e. Q to be the trivial form class) to make sure that $j(\mathfrak{k}), u(Q) \in \mathbb{R}$. We have previously determined that the ideal class group $\mathrm{Cl}(-4D)$, we copy what we have (4.17) and write it here again
$$\mathrm{Cl}(-4D) = C_2 \times C_{10} = \langle \mathbf{a} \rangle \times \langle \mathbf{b} \rangle$$
where $\mathbf{a} = [5, 0, 91]$ and $\mathbf{b} = [3, 2, 152]$. Moreover, in example 4.13, we find the order of the roots. Taking
$$H_0 = \langle e \rangle \quad \lhd \quad H_1 = \langle \mathbf{a} \rangle \quad \lhd \quad H_2 = \langle \mathbf{a} \rangle \times \langle \mathbf{b}^5 \rangle \quad \lhd \quad H_3 = \langle \mathbf{a} \rangle \times \langle \mathbf{b} \rangle.$$
as in (4.20) to be our composition series, we set $\mathcal{L}$ to be as in (4.15) and we have shown in that example that it is indeed an appropriate order to the elements of $\mathrm{Cl}(-4D) \cong G$. Applying u to the respective Q's in $\mathcal{L}$, we obtain the corresponding x_i's whose values are computed in example 4.16. Now, we begin the first iteration. We let
$$m = |H^{(1)}| = 2 \qquad \text{and} \qquad n = |G^{(1)}|/m = 20/2 = 10.$$
Example 4.18 already shows us which U_i's we need to compute. For completeness sake, we remark that
$$\mathcal{L}_0 = (e, \mathbf{b}^5, \mathbf{b}, \mathbf{b}^6, \mathbf{b}^2, \mathbf{b}^7, \mathbf{b}^3, \mathbf{b}^8, \mathbf{b}^4, \mathbf{b}^9)$$

[1] One can go about this by remembering the indices of the group elements of the form using a hash map to keep track of the respective inverse.

Taking $j = 1$, we compute the subproduct tree such that the root is a polynomial whose zeros are $(\vartheta_{i,j})_{0\leq i<n}$. For conjugate roots, we multiply them together to obtain quadratic polynomials with real coefficients. In doing this, we obtain the subproduct tree T in example 3.28. After checking, we see T_1 is irreducible. Finally, using the corresponding C (as discussed in (3.34)), we are able to compute W by computing $g_\ell(Y) = g_{\vartheta_{\ell,k}}$ for $\ell = 0, 1$, we obtain

$$g_1(Z) = 6Z^9 + 76Z^8 + 186Z^7 + 116Z^6 + 60Z^5 + 624Z^4 + 1414Z^3 + 1224Z^2 + 279Z - 110$$
$$g_0(Z) = -12Z^9 - 80Z^8 - 142Z^7 - 20Z^6 + 160Z^5 + 47Z^4 - 250Z^3 - 228Z^2 + 55Z + 95.$$

And so, we finally output

$$W_1 = X^2 - \frac{g_1(Y)}{V'(Y)}X + \frac{g_0(Y)}{V'(Y)}.$$

In preparation for the next iteration, we let

$$\mathcal{R}_V = (\vartheta_{i,k})_{0\leq i\leq n-1} = (x_{2i} + x_{2i+1})_{0\leq i\leq n-1}$$

and $\mathcal{L} = \mathcal{L}_1$. We now start the second iteration. This time around, we have

$$G^{(2)} = G/H_1 \qquad \text{and} \qquad H^{(2)} = H_2/H_1$$

and

$$m = |H^{(2)}| = 2 \qquad \text{and} \qquad n = |G^{(2)}|/m = 10/2 = 5.$$

We obtain

$$\mathcal{L}_2 = (e, \mathbf{b}, \mathbf{b}^2, \mathbf{b}^3, \mathbf{b}^4)$$

by taking every m^{th} element of $\mathcal{L}_1$. These are coset representatives for $G^{(2)}/H^{(2)}$. Since $\mathbf{b}, \mathbf{b}^4$ and $\mathbf{b}^2, \mathbf{b}^3$ are inverses, we only need to compute three U_i's:

i	elements of coset		U_i
0	$\{\mathbf{b}^0, \mathbf{b}^5\}$	$(X - x_0)(X - x_1)$	$X^2 - 10.47X + 1.774$
1	$\{\mathbf{b}^1, \mathbf{b}^6\}$	$(X - x_2)(X - x_3)$	$X^2 + (0.2277 + 1.228i)X + (-1.498 + 0.7888i)$
2	$\{\mathbf{b}^2, \mathbf{b}^7\}$	$(X - x_4)(X - x_5)$	$X^2 + (2.009 + 0.6565i)X + (1.111 + 0.9640i)$
3	$\{\mathbf{b}^3, \mathbf{b}^8\}$	$(X - x_6)(X - x_7)$	$\overline{U_2}$
4	$\{\mathbf{b}^4, \mathbf{b}^9\}$	$(X - x_8)(X - x_9)$	$\overline{U_1}$

Take note that in this iteration, x_i are the roots $\vartheta_{i,k}$ from the previous iteration. Taking $j = 1$ again and constructing the subproduct tree with $Z - 10.47$, $Z^2 + 4.018Z + 4.468$ and $Z^2 + 0.4555Z + 1.560$ as the leaves, we get

$$V^{(3)}(Z) = Z^5 - 6Z^4 - 39Z^3 - 74Z^2 - 80Z - 73.$$

And as before, we compute

$$W_2(Y) = Y^2 - \frac{g_1(Z)}{V'(Z)}Y + \frac{g_0(Z)}{V'(Z)}$$

where

$$g_1(Y) = 6Y^4 + 78Y^3 + 222Y^2 + 320Y + 365$$
$$g_0(Y) = Y^4 + 12Y^3 + 54Y^2 + 53Y - 140.$$

Since this is the final iteration, we let $W_3(Z) := V^{(3)}$. And so, we have that $W_1(X)$, $W_2(Y)$, $W_3(Z)$ are the polynomials which generate M/L_1, L_1/L_2 and L_2/K. That is, $L_2 = K[Z]/(W_3(Z))$, $L_1 = L_2[Y]/(W_2(Y))$ and $M = L_1[X]/(W_1(Z))$. We have solved problem 1.2.

Chapter 5

Complexity Analysis

An algorithm will not have much use if it could only handle small input in a reasonable amount of time. Hence, in this section, we analyze the time complexity of the algorithms we used. We wish to find an estimate on how long our algorithms are expected to run with respect to the size of the inputs. We start from the basic operations and work our way up to analyzing our more involved algorithms.

5.1 Polynomial Operations

We start by analyzing how many operations it takes to do addition and multiplication of univariate polynomials. Consider the polynomials $f, g \in K[X]$

$$f(X) = a_0 + a_1X + a_2X^2 + \ldots + a_dX^d$$
$$g(X) = b_0 + b_1X + b_2X^2 + \ldots + b_dX^d.$$

And assume their degree is at most d. Getting the sum f, g is very simple as you only need to add coefficients of the same power of X. This requires at most d operations in K. Hence, addition of polynomials takes $O(d)$ operations in K. As for multiplication, we know that

$$(f \cdot g)(X) = \sum_{k=0}^{2d} \sum_{i=0}^{k} a_k b_{k-i} X^k.$$

Multiplying the polynomials in this way requires at most d^2 additions and $(d+1)^2$ multiplications on K. From here, we see that multiplying two polynomials of degree at most d using this naive method takes at most $O(d^2)$ operations. We can do better than this! For simplicity of notation, assume $d = 2^k$ and observe that we can rewrite the polynomials as follows

$$\begin{aligned} f(X) \cdot g(X) &= \left(f_1(X) \cdot X^{d/2} + f_0(X)\right)\left(g_1(X) \cdot X^{d/2} + g_0(X)\right) \\ &= \underbrace{f_1(X)g_1(X)}_{p_2(X)} \cdot X^d + \underbrace{\left(f_1(X)g_0(X) + f_0(X)g_1(X)\right)}_{p_1(X)} \cdot X^{d/2} + \underbrace{f_0(X)g_0(X)}_{p_0(X)} \end{aligned}$$

where $\deg f_1, \deg f_0, \deg g_1, \deg g_0 \leq d/2$. At first glance, it seems that we need to multiply four pairs of polynomials. We note Karatsuba's observation that

$$p_1(X) = \left(f_1(X) + f_0(X)\right)\left(g_1(X) + g_0(X)\right) - p_2(X) - p_0(X).$$

And so, after making two multiplications to obtain $p_2(X)$ and $p_0(X)$, we are only required to make one additional multiplication to obtain $p_1(X)$. We denote by $M_X(d)$ to be the number of multiplications in K to multiply two polynomials in $K[X]$ of degree at most d. Since all multiplications required to obtain the p_i's involve polynomials of degree $d/2$, we see that we need at most time $3M_X(d/2)$ to multiply two polynomials of degree at most d. Unraveling the recursion, see that it takes $3^k = d^{\log_2 3} \approx d^{1.58}$ multiplications in K. This is an improvement to our earlier naive approach. In fact, we can do even better if the field K has a d^{th} root of unity ω. We can multiply two polynomials F, G using the Discrete Fast Fourier transform (DFT). Fix ω is a d^{th} root of unity, we define the Fourier transform of a vector $\mathbf{t} = (t_1, \ldots, t_d)$ to be

$$\mathcal{F}_\omega(\mathbf{a}) = (T(\omega^0), T(\omega^1), \ldots, T(\omega^{d-1}))$$

where

$$T(X) = t_0 + t_1 X + t_2 X^2 + \ldots + t_{d-1} x^{d-1}. \tag{5.1}$$

It can be computed by the following algorithm:

Algorithm 5.2. `discreteFFT`

INPUT: T with $\deg T < d = 2^k$ and $(\omega^0, \ldots, \omega^{d-1})$.

OUTPUT: $\mathcal{F}_\omega(T)$

ALGORITHM:

1. If $\deg T < 1$, output the vector T.
2. Compute $(a_0, \ldots, a_{n/2-1}) = \mathcal{F}(T_0, \omega^2)$ and $(b_0, \ldots, b_{n/2-1}) = \mathcal{F}(T_1, \omega^2)$ where $T_0(X^2) = \frac{T(X)+T(-X)}{2}$ and $XT_1(X^2) = \frac{T(X)-T(-X)}{2}$.
3. Return $(a_i + \omega^i b_i)_{i=0,\ldots,d-1}$.

Theorem 5.3. Algorithm 5.2 requires $O(d \log d)$ operations in K.

Proof. If we take $f(t)$ to be the number of operations it takes for algorithm 5.2 to finish, we have that

$$f(t) \leq f(t/2) + f(t/2) + O(t)$$

where the first and second term are for computing step 2 and $O(t)$ is for computing step 3. Since the algorithm is going to be called approximately $\log 2t$ times, we get that $f(t) = O(t \log t)$ □

Moreover, by following the definition, we have the following property

$$\mathcal{F}_{\omega^{-1}}(\mathcal{F}_\omega(T)) = dT.$$

If $\deg F + \deg G < d$, and writing

$$F(X) = a_0 + a_1 X + a_2 X^2 + \ldots + a_{d-1} X^{d-1}$$
$$G(X) = b_0 + b_1 X + b_2 X^2 + \ldots + b_{d-1} X^{d-1}.$$

It can be shown that the product of F and G is

$$(FG)(X) = c_0 + c_1 X + c_2 X^2 + \ldots + c_{d-1} X^{d-1} \tag{5.4}$$

where

$$(c_0, \ldots, c_{N-1}) =: c = \frac{1}{d}\mathcal{F}^{-1}(\mathcal{F}(\mathbf{a})\mathcal{F}(\mathbf{b})).$$

To obtain (5.4), we need to apply $\mathcal{F}$ three times to vectors of size d and then divide the everything by d.

Corollary 5.5. Multiplication using FFT requires $O(d \log d)$ operations in K where d is the maximum degree of the polynomials.

5.2 Floating Point Operations

A floating point number consists of an η-bit integer a and an integer exponent e and this represents the rational number $a \cdot 2^e$, which we use to approximate real numbers. Multiplying two floating point numbers is as easy as multiplying two η-bit integers. Viewing η-bit integers as degree η polynomials whose coefficient is 2, we see by Karatsuba's method that it should take at most $O(\eta^{\log_2 3})$ bit operations to multiply two integers. There is an algorithm by Schönage-Strassen [9], which is based on the DFT and can bring down the required bit operations to $O(\eta \log \eta \log\log \eta)$. Adding two floating point numbers of the same exponent is just the same as adding two integers and hence it takes $O(\eta)$ bit operations. If the exponents, however are not the same, we repeatedly divide the a by 2 and then add 1 to e of the smaller number until the exponents are equal. If the difference of the exponents are greater than η, then this number becomes 0 even before we catch up with the exponent of the other number. Thus, adding floating point numbers takes $O(\eta)$ bit operations.

5.3 Analyzing one iteration

Now, we are ready to analyze the time complexity of constructing the subproduct tree.

Theorem 5.6. Let s be the number of leaves in the input of 3.25. Assume that the leaves are linear polynomials. Then algorithm 3.25 requires $O(M_X(s)\log(s))$ operations in $\mathbb{C}$.

Proof. Let $f(t)$ be the time it takes to compute the T_1' of a subtree T' with t leaves. We have

$$f(t) = f(t/2) + f(t/2) + O(M(t))$$

since to compute the top (i.e. the root) of a subtree with t leaves, one must compute the values of its two children (accounting for the $f(t/2)$'s) and multiply them. Since they are of degree $t/2$, then we expect the multiplication to take time $O(t \log t)$. And so to computing the T_1 of a tree with s leaves takes $O(\log s \cdot M_X(s))$ oeprations in $\mathbb{C}$. □

Algorithm 3.29 has similar steps as 3.25 except for the last one, where instead of multiplying one pair of polynomials in the end, we perform two multiplications. This does not change the asymptotic complexity and so we have that.

Theorem 5.7. Algorithm 3.29 has requires $O(M_X(s)\log s)$ operations in $\mathbb{C}$ where s is the number of leaves on the tree T.

Corollary 5.8. If n is the degree of the extension L, then computing one g_ϑ using algorithm 3.31 requires $O(M(n)\log n)$ operations in $\mathbb{C}$.

Proof. Algorithm 3.31 simply calls algorithms 3.25 and 3.29. □

Remark 5.9. We make a remark on the time complexity of the following steps of the main algorithm 4.19:

- Finding the n U_i's involves building n subproduct trees with m leaves. This means that it would take time $O(n \cdot M_X(m) \log m)$ operations in $\mathbb{C}$ to compute all U_i's. We can reduce the computatons by a factor of at most 2 if we only compute for U_i's which are in $\mathcal{I}$. However, this does not change the asymptotic complexity.
- Finding the appropriate V involves building at most m subproduct trees with n leaves. Thus, it requires $O(m \cdot M_X(n) \log n)$ operations in $\mathbb{C}$. It might be useful to note that in practice, taking the trace of the polynomial (i.e. the case where $j = m - 1$) results in an irreducible V.
- Moreover, checking the irreducibility be looking if $\gcd(V, V') \neq 1$ using the fast Euclidean algorithm from [9] takes $O(M(n) \log n)$ operations in $\mathbb{C}$.
- Finding W takes $O(m \cdot M_X(n) \log n)$ operations in $\mathbb{C}$ because you need m calls to algorithm 5.8.

Now, the only iterating step left unaccounted for is the second step. This step actually does not make a dent in the asymptotic complexity as those computations do not involve complex numbers, but integers, and these integers are relatively small compared to the precision required for the other bigger computations. Hence, we conclude this section with the following theorem.

Theorem 5.10. One iteration of the main algorithm 4.19 requires

$$O(mM_X(n) \log n + nM_X(m) \log m)$$

computations in $\mathbb{C}$.

5.4 Total complexity

So far, we have postponed taking into account a couple of things for the main algorithm 4.19. First, we did not look at the expected number of iterations. Denote t by the number of primes in the prime decomposition (counting multiplicity) of the class number h_K of K. The number of iterations is exactly $t - 1$. Second, we did the complexity analysis by counting the operations in $\mathbb{C}$. To solve for the number of bit operations, we need to find the appropriate precision η. Recall that U_0 and V must be polynomials in $\mathbb{Z}[X]$. Since we compute V in algorithm 4.19 as the product of linear polynomials, it will surely be reducible if we do not round it into a polynomial in $\mathbb{Z}$. This is a step which is sensitive to precision. One can find a lower bound for η. To be precise, we refer to the theorem in [5]. To conclude, we have the following corollaries to 5.10.

Corollary 5.11. Assuming GRH and using the DFT methods for multiplication, one iteration in algorithm 4.19 requires

$$O\left(\sqrt{D}(\log D)^2 \log\log D\right).$$

operations in $\mathbb{C}$. That is, it requires

$$O\left(D(\log D)^3(\log\log D)^2\right)$$

bit operations.

Proof. We note that $h = mn$, where h is the class number. From theorem 5.10, we get that we require

$$O(h \log^2 h)$$

operations on each of the t iterations. And since under GRH, [5] tells us that $h = O(\sqrt{D} \log\log D)$, we need

$$O(h(\log D)^2) = O\left(\sqrt{D}(\log D)^2 \log\log D\right)$$

computations in $\mathbb{C}$ for each iteration. We compute the bit operations. From [5],we have that

$$M(\eta) = O(\sqrt{D} \log D \log\log D),$$

we get that we need

$$O\left(D(\log D)^3(\log\log D)^2\right)$$

bit operations for each iteration. □

Corollary 5.12. Assuming GRH and using the DFT methods for multiplication, algorithm 4.19 requires

$$O\left(\sqrt{D}(\log D)^3 \log\log D\right).$$

operations in $\mathbb{C}$. That is, it requires

$$O\left(D(\log D)^4(\log\log D)^2\right)$$

bit operations.

Proof. Since $t = O(\log h) = O(\log D)$, then from the previous corollary 5.11, we get the desired results. □

Chapter 6

Verifying the Results

We have proved that everything should work in theory. However, translating theory into an implementation is a different story. For the interested readers, in this section, we present some ways to determine if something is wrong with the implementation.

6.1 Using resultants

Definition 6.1. Let R be an integral domain. Let $f, g \in R[X]$. Writing

$$f(X) = \prod_{i=1}^{m} a(X - \alpha_j) \qquad \text{and} \qquad g(X) = \prod_{j=1}^{n} b(X - \beta_j)$$

with α_i's and β_j's roots of f and g in an algebraic closure of the field of fractions of R. We define the resultant of f and g to be

$$\mathrm{Res}(f, g) = a^n b^m \prod_{i=1}^{m} \prod_{j=1}^{n} (\alpha_i - \beta_j).$$

Definition 6.2. Let K be a field and let $f, g \in K[X, Y]$. Define the resultant $\mathrm{Res}_X(f, g)$ of f and g with respect to X to be the resultant of f and g considered as polynomials in X. Similarly, we can define $\mathrm{Res}_Y(f, g)$.

Example 6.3. Let $V(Y) \in K[Y]$ and $W(X) \in K(Y)[X]$. By clearing denominators, we can replace W by a polynomial in $K[X, Y]$. From now on, we assume $W(X, Y) \in K[X, Y]$. Consider $\mathrm{Res}_Y(V, W)$. It can be shown that this is a polynomial in $K[X]$. Moreover, if β is a root of $\mathrm{Res}_Y(V, W)$, that is $\mathrm{Res}_Y(V, W)(\beta) = 0$, then the polynomials $V(Y)$ and $W(\beta, Y)$ have a common root α. Hence, we have that $V(\alpha) = W(\beta, \alpha)$.

Theorem 6.4. Let M/K be a finite abelian extension. Let $V(Y) \in K[Y]$ and $W(X) \in L[X]$ be minimal polynomials of the extensions L/K and M/L, respectively. Let α be a root of V. Using lemma 3.22, we can write W as

$$W(X) = \frac{1}{V'(\alpha)} \sum_{i=0}^{m} g_i(\alpha) X^i$$

where $g_i(Y) \in K[Y]$. If

$$\tilde{W}(X, Y) = \sum_{i=0}^{n} g_i(Y) X^i,$$

then $\mathrm{Res}_Y(V, \tilde{W})$ generates the field extension M/K.

Proof. Suppose α and β is a root of $V(Y)$ and β is a root of $\tilde{W}(X,\alpha)$. Then, $V(\alpha) = W(\beta,\alpha) = \tilde{W}(\beta,\alpha) = 0$. This means that β is a root of $\mathrm{Res}_Y(V,\tilde{W})$. Suppose $n := \deg V$ and $m := \deg W$. Note that V has $m := \deg V$ distinct roots. Fixing an α and considering W_α, we obtain $n := \deg_X \tilde{W}$ distinct roots. If β is one of these roots, then the rest are $(\beta^\sigma)_{\sigma\in\mathrm{Gal}(M/L)}$. Hence, the roots of $\mathrm{Res}_Y(V,\tilde{W})$ are exactly $(\alpha^\tau)^\sigma$ where τ runs through $\mathrm{Gal}(L/K)$ and σ runs through $\mathrm{Gal}(M/L)$. Hence, $\mathrm{Res}_Y(V,\tilde{W})$ is an irreducible polynomial of degree mn. And since $[M:K] = mn$, it is a minimal polynomial for the extension M/K. □

6.2 Constructing an elliptic curve of good cardinality

The theory on resultants only checks if the field was successfully decomposed given a polynomial h_K which supposedly generates M/K. However, we have no way to check if h_K indeed generates the Hilbert class field – that is $M = H_K$. Remember that we computed h by first getting the roots. If there was some problem in the implementation of our code, we would not be able to detect errors in our computation. Another way to check if our implementation is correct is to actually generate a curve with a prescribed number of points using algorithm 1.1. Not only does this give us a way to determine if our implementation is wrong, it also gives us an illustration of what our main algorithm, algorithm 4.19, can be applied to.

Let H_K be the Hilbert class field and take γ to be a root of the Hilbert class polynomial h_K as a polynomial in $\mathbb{Z}$. Then $H_K = K(\gamma)$. Finding a root $\tilde{j}$ of h_K modulo p is equivalent to finding a linear factor $f_{\tilde{j}} = (X - \tilde{j})$ of h_K modulo p. By the theorem of Kummer-Dedekind, finding such a linear factor is equivalent to finding a prime ideal $\mathfrak{P} = (p, f_{\tilde{j}}(\gamma))$ of degree 1 above (p). Now, since H_K is an unramified extension of K, there will exist exactly $h = \deg h_K$ such prime ideals $\mathfrak{P}$, each corresponding to a root of h_K modulo p. Now, suppose we have an intermediate field $L = K(\alpha)$. Let V and W respectively be the minimal polynomials of L/K and H_K/L. Then, in a similar way, taking a root of V modulo p will give us a prime ideal $\mathfrak{p} = (p, V(\alpha))$ of L. Now, $W(X) \in \mathcal{O}_L[X]$ and taking it modulo $\mathfrak{p}$, we get that it is in $W(X) \in (\mathcal{O}_L/\mathfrak{p})[X]$. Since we are working in an unramified extension and the degree of $\mathfrak{P}$ in M/K is 1, $\mathcal{O}_L/\mathfrak{p} \cong \mathbb{F}_p$. And so finding a linear factor of $W(X)$ as a polynomial in $\mathbb{F}_p$ will give us a prime ideal in $\mathfrak{P}'$, which corresponds to a root of h_K.

Since we have completely avoided dealing with the Hilbert class polynomials in the examples, and instead focused on the Weber polynomials, we must find a method on how to transform a root of a Weber class polynomial into a root of the Hilbert class polynomial. The following theorem from [7] takes care of this

Theorem 6.5. Let D be a fundamental discriminant such that $D \not\equiv 0 \pmod 3$. Let $\mathscr{W}_K$ be the corresponding Weber polynomial and h_K be the corresponding Hilbert class polynomial for K, both of which generate H_K. If w is a root of the $\mathscr{W}_K$, then

$$j = \frac{(A-16)^3}{A}$$

is a root of h_K where

$$A = \begin{cases} 2^{12}w^{-24} & \text{if } D \equiv 3 \pmod 8 \\ w^{-24} & \text{if } D \equiv 7 \pmod 8 \\ -2^6 w^{12} & \text{if } D/4 \equiv 2, 6 \pmod 8. \end{cases}$$

Example 6.6. Suppose we want to find an elliptic curve E with exactly $N = 380$ points over the finite field $\mathbb{F}_p$ with $p = 491$. We first observe that

$$-455 \cdot 4 = -1820 = (p + 1 - N)^2 - 4p = 144 - 1964.$$

Recall that in example 4.20, we decomposed the Weber polynomial $\mathscr{W}_K$ for $K = \mathbb{Q}(\sqrt{-455})$. Solving $W_3(Z)$ as a polynomial modulo p, we see that $z = 406$ is one of its roots modulo p. Substituting z into $W_2(Z, Y)$, we obtain the polynomial

$$Y^2 + 85Y + 12 \in \mathbb{F}_p[Y].$$

One of the roots of this polynomial is $y = 477$. Again, substituting into $W_1(Y, X)$, we obtain the polynomial

$$X^2 + 14X + 447 \in \mathbb{F}_p[X].$$

Finally, a root of this polynomial is $w = 382$. Hence w is a root of the Weber polynomial. Applying 6.5, we take

$$j = \frac{(w^{-24} - 16)^3}{w^{-24}} = 139.$$

Let

$$a = \frac{27j}{4(1728 - j)} = 352$$

We take the elliptic curve

$$E : Y^2 = X^3 + 352X - 352$$

over the finite field $\mathbb{F}_p$. And this curve, indeed, has 480 points.

Chapter 7

Experiments

We implement the main algorithm 4.19 in Pari/GP [2]. We run the implementation through discriminants D such that $D \equiv 7 \pmod 8$, $D \not\equiv 0 \pmod 3$ and $D < 1000000$. Out of the 76002 fundamental discriminants which satisfies the above conditions, 1 of them correspond to discriminants where $K = \mathbb{Q}(\sqrt{-D})$ has class number 1. Moreover, 5256 of them correspond to cases where K has prime class number. We do not apply the algorithm to these cases since we will not be able to find intermediate fields. Hence, we treat a total of 70745 cases. To make the computation faster, we let the precision η grow in terms of the discriminant D and the class number h. We take $\eta = \frac{1}{9} h_D \log(D)$ althroughout. Though one could readjust to a lower precision after each iteration.

With this, the slowest case was when $D = 947231$, which took 5.382 seconds. In this case, the class number $h_D = 1766 = 883 \times 2$. This means that we had to obtain $m = 883$ quadratic polynomials to get and eventually find 883 Hecke representations. In contrast, the slowest case without a big prime is $D = 905399$, which has a class number $h_D = 1536 = 2^9 \times 3$. Take note that this has class number close to h_{947231}. However, this takes only 2.087 seconds to compute.

7.1 On the irreducibility of the V_k

We notice that the first V_k we usually take, that is V_{m-1}, whose roots are the traces of the U_i, almost always results in an irreducible polynomial. Indeed, among the D we treated, here are the only five cases where the trace does not work.

D	V_{m-1}
55	$x^2 - 2x + 1 = (x-1)^2$
95	$x^4 - 2x^3 - x^2 + 2x + 1 = (x^2 - x - 1)^2$
119	$x^2 - 4x + 4 = (x-2)^2$
239	$x^2 - 4x + 4 = (x-2)^2$
287	$x^2 - 8x + 16 = (x-4)^2$

Note that in general these polynomials depend on how we chose the composition series. A different composition series might result in a different set of "problematic" cases.

7.2 On the composition series

In this section, we see that choosing an appropriate order for the composition series is important. We illustrate this by applying looking at the case when $D = 928919$. We compute

$$h_D = 1534 = 2 \times 13 \times 59$$

and the class group $\mathrm{Cl}(\mathcal{O}_K)$, which turns out to be cyclic. Let its generator be $\mathbf{a}$. We consider two composition series

$$H_0 = \langle e \rangle \quad \lhd \quad H_1 = \langle \mathbf{a}^{26} \rangle \quad \lhd \quad H_2 = \langle \mathbf{a}^2 \rangle \quad \lhd \quad H_3 = \langle \mathbf{a} \rangle \tag{7.1}$$

and

$$H'_0 = \langle e \rangle \quad \lhd \quad H'_1 = \langle \mathbf{a}^{767} \rangle \quad \lhd \quad H'_2 = \langle \mathbf{a}^{13} \rangle \quad \lhd \quad H'_3 = \langle \mathbf{a} \rangle. \tag{7.2}$$

Using the composition series given by (7.1), we have $m^{(1)} = |H_1| = 59$. And so, for the second iteration has to deal with a group of size $|G^{(2)}| = 26$. On the other hand, in the compositin series given by (7.2), $m^{(1)} = |H'_1| = 2$. And the second iteration will involve a group of size $|G^{(2)}| = 767$. Hence, looking at the asymptotic complexity of the algorithm, we expect that using a composition series such as (7.1) will start to benefit us starting from the second iteration. However, looking at the running time, we have

composition series used	ι	$\lvert G^{(i)} \rvert$	time (ms)
(7.1)	1	1534	492
(7.1)	2	26	4
(7.2)	1	1534	2191
(7.2)	2	767	208

Looking at the running time, not only do we benefit from the second iteration, but we also benefit from the iteration. This is because computing W takes a lot of time in practice. So we prefer to have a lower degree for W, which is done by taking a lower m.

And so, we should choose the composition series such that the sequence of $m^{(\iota)} = |H^{(\iota)}|$ we choose decrease as ι increases. Doing so makes the $n^{(\iota)}$ of the next cases decrease faster. Note that this was not done in example 4.20 for illustrative purposes.

Bibliography

[1] Atkin, A. O. L., and Morain, F. Finding suitable curves for the elliptic curve method of factorization. *Mathematics of Computation 60*, 201 (Jan 1993), 399–405.

[2] Belabas et al., K. Pari/GP, 2.8.0 ed. Bordeaux, 2015. `http://pari.math.u-bordeaux.fr/`.

[3] Bröker, R. M. *Consturcting Elliptic Curves of Prescribed Order*. PhD thesis, Universiteit Leiden, jun 2006.

[4] Cox, D. A. *Primes of the Form x + ny: Fermat, Class Field Theory, and Complex Multiplication*. John Wiley Sons, Inc. All rights reserved., 1989.

[5] Enge, A. The complexity of class polynomial computation via floating point approximations. *Mathematics of Computation* (2009), pp. 10891107.

[6] Enge, A., and Morain, F. Fast decomposition of polynomials with known galois group. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes – AAECC-15* (2003), T. H. In Marc Fossorier and A. Poli, Eds., vol. 2643 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 254264.

[7] Konstantinou, E., Stamatiou, Y., and Zaroliagis, C. On the use of weber polynomials in elliptic curve cryptography. In *European PKI Workshop* (2004), Lecture Notes in Computer Science, Springer Verlag, pp. 335–349.

[8] Schertz, R. *Complex multiplication*. New Mathematical Monographs. CUP, 2010.

[9] von zur Gathen, J., and Gerhard, J. *Modern computer algebra*, 1 ed. Cambridge University Press, 1999.