CSE291-14: The Number Field Sieve

https://cseweb.ucsd.edu/classes/wi22/cse291-14

Emmanuel Thomé

March 3rd, 2022

Part 8b

Virtual and individual logarithms

Plan

Plan

What do we need?

Algebraic structure of the set of $(a - b\alpha_0)$:

- (abelian) subgroup of $\mathbb{Q}(\alpha_0)^\times$.
- $\mathbb{Z}$-module, if we so wish.
- It is also finitely generated, thanks to smoothness. (We just proved it!)

If we look at this set of $(a - b\alpha_0)$ modulo ℓ-th powers, we see that it is a finite dimensional $\mathbb{Z}/\ell\mathbb{Z}$-vector space.

Vector spaces

$$\begin{tikzcd}
& \text{lucky elements in group generated by } \mathbb{Z}[x] - \langle p, x-m \rangle \arrow[dl, "x \to \alpha_0"'] \arrow[dr, "x \to \alpha_1"] & \\
(\text{FDVS}) \quad \Sigma_0 = \text{subgroup of } \mathbb{Q}(\alpha_0)^\times \text{ mod } \ell\text{-th powers} \arrow[dr, "\alpha_0 \to m"] \arrow[ddr, bend right, "(\text{curvy map})_0"'] & & \text{subgroup of } \mathbb{Q}(\alpha_1)^\times \text{ mod } \ell\text{-th powers} = \Sigma_1 \quad (\text{FDVS}) \arrow[dl, "\alpha_1 \to m"'] \arrow[ddl, bend left, "(\text{curvy map})_1"] \\
& (\mathbb{F}_p^\times)/(\mathbb{F}_p^\times)^\ell \text{ also FDVS} \arrow[d, "\mathbf{L}"] & \\
& \mathbb{Z}/\ell\mathbb{Z} &
\end{tikzcd}$$

Our goal (as stated): find **L**.

Alternative goal: find the two curvy linear forms, + a lifting map.

Previous picture is important

Digression: how can we characterize elements in Σ_0 and Σ_1?

Note: in the following slides, α, K, Σ denote things that can be instantiated on either side of the diagram.

Plan

Riddles

I'm thinking of some rational number.

What does it take to identify it without ambiguity?

- valuations (in $\mathbb{Z}$) at all primes.
- sign.

Riddles

I'm thinking of some B-smooth rational number.

What does it take to identify it without ambiguity?

- valuations (in $\mathbb{Z}$) at all primes below B.
- sign.

Riddles

I'm thinking of some B-smooth rational number mod ℓ-th powers.

What does it take to identify it without ambiguity?

- valuations (in $\mathbb{Z}/\ell\mathbb{Z}$) at all primes below B.
- ~~sign~~. (not needed since ℓ odd).

Next: we want to generalize.

Characterizing an algebraic number

I'm thinking of some element of Σ (Σ_0 or Σ_1).
i.e. some B-smooth element of $\mathbb{Q}(\alpha)^\times$ mod ℓ-th powers.

To identify it without any ambiguity, we need:

- valuations modulo ℓ at all prime ideals whose norm is $\leq B$.
- any information that can break ties. Ties are related to units.
  - in our rosy picture, valuations at the generating units played this role.
  - but in fact, we do not have to take this long-winded path. Other functions could do.

Characterizing with characters

Requirements for tie breakers

Suppose that we have a finite set of characters $\{\chi\}$ such that:

- The domain of all χ is $\mathbb{Q}(\alpha)^\times$ (focus: *B*-smooth elements).
- $\chi : (\mathbb{Q}(\alpha)^\times, \times) \to (\mathbb{Z}/\ell\mathbb{Z}, +)$.
- Characters χ are efficiently computable.
- $\mathcal{O}_K^\times \cap \left(\bigcap_\chi \operatorname{Ker} \chi\right) = (\mathcal{O}_K^\times)^\ell$.

Then two elements that agree on all ideal valuations AND on all of these characters must be equal modulo ℓ-th powers.

In the NFS-DL context, we use the Schirokauer maps for these characters.

Valuations and characters

Consider the set $\{\nu_{\mathfrak{p}}\}_{N(\mathfrak{p})\leq B} \cup \{\chi_1, \ldots, \chi_c\}$.

- $\nu_{\mathfrak{p}}$ are valuations $\bmod \ell$ at each ideal that is allowed to appear.
- χ_i are our characters.

All of these are linear forms defined on Σ.

As such, they are elements of the dual space Σ^*.

The curvy map $\Sigma \to \mathbb{Z}/\ell\mathbb{Z}$ that we are looking for is also an element of Σ^*.

Plan

Virtual logarithms

If $\mathbf{B} = \{\nu_{\mathfrak{p}}\} \cup \{\chi_1, \ldots, \chi_c\}$ is a basis of the dual space Σ^*.
(analogue for $\mathbb{Q}$: elements of $\mathbf{B}$ would be just "valuations at primes")

- Then the curvy map $\Sigma_0 \to \mathbb{Z}/\ell\mathbb{Z}$ decomposes along $\mathbf{B}_0$.
  $=$ There exists an identity of linear forms:

$$(\text{curvy map})_0 = \sum_{e \in \mathbf{B}_0} \lambda_e e.$$

- Because $a - b\alpha_0$ and $a - b\alpha_1$ are known to map identically through $(\mathbb{F}_p^\times)/(\mathbb{F}_p^\times)^\ell$, then both curvy maps must agree:

$$(\text{curvy map})_0(a - b\alpha_0) = (\text{curvy map})_1(a - b\alpha_1).$$

$$\sum_{e \in \mathbf{B}_0} \lambda_e e(a - b\alpha_0) = \sum_{e' \in \mathbf{B}_1} \lambda_{e'} e'(a - b\alpha_1).$$

Virtual logarithms

$$\sum_{e \in \mathbf{B}_0} \lambda_e e(a - b\alpha_0) = \sum_{e' \in \mathbf{B}_1} \lambda_{e'} e'(a - b\alpha_1).$$

Virtual logarithms

The virtual logarithm of $e \in \mathbf{B}$ is the coefficient λ_e.

- It can be the virtual logarithm at some ideal, or the virtual logarithm at some character.
- Relations in NFS-DL express necessary conditions on all these virtual logarithms.
- As previously mentioned, our linear system finds solutions up to a constant factor, since all constraints are homogeneous.

The “too many characters” case

Other case: $\mathbf{B} = \{\nu_{\mathfrak{p}}\} \cup \{\chi_1, \ldots, \chi_c\}$ contains a basis of the dual space Σ^*.

- Then there is some ambiguity in the decomposition along $\mathbf{B}_0$.
- This means that a full vector space of (all “correct”) virtual logarithm choices work. None is privileged.

Bear in mind

Virtual logarithm = the coefficient that goes alongside a "valuation at $\mathfrak{p}$" or "character value at χ" and meets the conditions that come from the diagram.

If everything were simple and easy, we would only have valuations at a set of elements, and these coefficients would be logarithms of images of these elements in $\mathbb{F}_p$.

Linear algebra

The linear algebra problem

A pair (a, b) yields a constraint that binds the virtual logarithms.
With sufficiently many constraints, we have the correct kernel.
Equations in this linear system are hybrid:

- Coefficients $\nu_{\mathfrak{p}}(a - b\alpha)$ come from valuations (very sparse), and coefficients $\chi(a - b\alpha)$ are likely to be dense.
- The system can be written as $M \times v = 0$, or $(M|C) \times v = 0$. (depending on whether we include the character block C in M or not.)

This is homogeneous, and we are looking for a right kernel vector.

This is subtly different from the factoring case.

Our characters here have to enter before linear algebra.

Plan

Requirements

Requirements for characters $\Sigma \to \mathbb{Z}/\ell\mathbb{Z}$

We want a finite set of characters $\{\chi\}$ such that:

- The domain of all χ is $\mathbb{Q}(\alpha)^\times$ (focus: *B*-smooth elements).
- $\chi : (\mathbb{Q}(\alpha)^\times, \times) \to (\mathbb{Z}/\ell\mathbb{Z}, +)$.
- Characters χ are efficiently computable.
- $\mathcal{O}_K^\times \cap \left(\bigcap_\chi \operatorname{Ker} \chi\right) = (\mathcal{O}_K^\times)^\ell$.

We will build such characters with ℓ-adic arithmetic.

Setting

We can safely assume that:

- we deal with elements that are coprime to $\ell\mathcal{O}_K$. (B-smooth elements and their products certainly are).
- the ideal $\ell\mathcal{O}_K$ does not ramify (ℓ is coprime to $\operatorname{disc} K$), and ℓ is coprime to f_d as well.

Let us suppose that the defining polynomial factors modulo ℓ as

$$f(x) \mod \ell = (\text{degree } d_1) \times \cdots \times (\text{degree } d_k).$$

(with no repeated factors, by our assumption above.)

Step 1: $\Sigma \to \mathcal{O}_K/\ell\mathcal{O}_K$

The quotient $\mathcal{O}_K/\ell\mathcal{O}_K$ is a product of fields, not a field.

The projection is a simple coefficient-wise reduction.

$$R : \left\{ \begin{array}{rcl} \mathcal{O}_K & \to & \mathcal{O}_K/\ell\mathcal{O}_K \cong \mathbb{F}_{\ell^{d_1}} \times \cdots \times \mathbb{F}_{\ell^{d_k}} \\ \phi(\alpha), \phi \in \mathbb{Z}[x] & \mapsto & \phi(R(\alpha)), \phi \in \mathbb{Z}/\ell\mathbb{Z}[x]. \\ u/v & \mapsto & R(u)/R(v). \end{array} \right.$$

Elements coprime to $\ell\mathcal{O}_K$ map to invertible elements, so the projection R is well defined on Σ as well.

For any element $u \in \Sigma$:

- the first component of $R(u)$ is in $(\mathbb{F}_{\ell^{d_1}})^\times$. Then its $(\ell^{d_1} - 1)$-th power is 1.
- by extension, the $\mathrm{lcm}(\{\ell^{d_i} - 1\}_i)$-th power of $R(u)$ is 1 on all components, so it is really 1.

Precision ℓ^2

To obtain something useful, we need R to project modulo ℓ^2 instead of just ℓ.

Schirokauer maps

Let $R : \mathcal{O}_K \mapsto \mathcal{O}_K/\ell^2\mathcal{O}_K$.
Let $X_u(x)$ be the degree-$(d-1)$ polynomial such that:

$$R(u(\alpha))^{\operatorname{lcm}(\{\ell^{d_i}-1\}_i)} = 1 + \ell \cdot X_u(R(\alpha)) \mod \ell^2.$$

What is X_{uv}? Since $(1+\ell x)(1+\ell y) = (1+\ell(x+y)) \mod \ell^2$, we have:

$$X_{uv} = X_u + X_v.$$

Therefore $u \mapsto X_u$ is a morphism from $(\Sigma, \times)$ to $(\mathbb{Z}/\ell\mathbb{Z}^d, +)$.

Sage code

```
ZP.<x>=ZZ['x']
f=ZP([randrange(-100,100) for i in range(6)])
ell=1009
assert gcd(ell,f.discriminant()*f[f.degree()]) == 1
f2=ZP.change_ring(R2)(f)
ZR2ell=quotient(ZP.change_ring(R2),f2/f2[f2.degree()])
big=lcm([ell^(fac[0].degree())-1
    for fac in f.change_ring(GF(ell)).factor()])
def schirokauer_map(u):
    # u a polynomial with coefficients in ZZ
    return vector([GF(ell)(ZZ(c)//ell)
        for c in list(ZR2ell(u)^big-1)])
```

Sage code

```
sage: u=ZP([randrange(10) for i in range(5)])
sage: v=ZP([randrange(10) for i in range(5)])
sage: schirokauer_map(u)
(772, 522, 920, 969, 740)
sage: schirokauer_map(u*v)
(415, 10, 406, 215, 473)
sage: schirokauer_map(u)+schirokauer_map(v)
(415, 10, 406, 215, 473)
```

Do we meet the requirements?

If units are well-behaved, yes.

- In some highly unlikely scenarios, we might need larger ℓ-precision.
- Some more dramatic failures are guarded by Leopoldt's conjecture (widely believed to be true).

In all likelihood, if u is a unit, then $X(u) = 0$ iff $u \in (\mathcal{O}_K^\times)^\ell$.

The set of coordinates of X can be used as our desired set of characters $\{\chi\}$.

Faster computation

It is actually a little bit faster to compute the Schirokauer maps piecewise modulo each of the factors.

We may decide to compute only part of the coordinates, if we know that the unit rank is not that big.

Plan

The linear algebra problem

Values of the Schirokauer maps

If we write the linear system as $(M \mid C) \times v = 0$, note that:

- M: very sparse, small coefficients.
- C: dense, coefficients modulo ℓ.

Note that we cannot first ignore the C block, and add it afterwards. It won't work. So the factoring case $(v \times M)$ and DLP case $(M \times v)$ are very different.

First approach

The first approach is to solve the system $(M \mid C) \times v = 0$ as is.

- Each matrix-times-vector operation involves multiplication of several full-length integers modulo ℓ.
    - Typically, "normal" matrix coefficients are ± 1, and processing them entail one addition modulo ℓ.
    - Multiplications cost $O(\log \ell)$ times more, and dealing with the Schirokauer block ends up being a large part of the overall cost.
- This also means more code, more storage, etc.
  DLP240: $M \approx 70G$, $C \approx 30G$.

Note that in this context, one kernel vector is enough.

Alternative approach

It is way better to use the block Wiedemann in a smart way.

Assumptions:

- r columns in the Schirokauer block,
- BW blocking parameters m, n such that $n \geq r$.
  For simplicity, assume $n = r$.

Use only M as the matrix, and in BW set $y = C$.

Given one column j of the linear generator matrix, we have:

$$\begin{aligned} 0 = &(C_0 f_{0,j,0} + \cdots + C_{r-1} f_{r-1,j,0}) + \\ &M(C_0 f_{0,j,1} + \cdots + C_{r-1} f_{r-1,j,1}) + \\ &M^2(C_0 f_{0,j,2} + \cdots + C_{r-1} f_{r-1,j,2}) + \cdots \end{aligned}$$

Alternative approach

$$\begin{aligned} 0 = {} & (C_0 f_{0,j,0} + \cdots + C_{r-1} f_{r-1,j,0}) + \\ & M(C_0 f_{0,j,1} + \cdots + C_{r-1} f_{r-1,j,1}) + \\ & M^2(C_0 f_{0,j,2} + \cdots + C_{r-1} f_{r-1,j,2}) + \cdots \\ 0 = {} & C \times \text{some vector} + M \times \text{some other vector}, \end{aligned}$$

which is exactly what we're looking for. In particular, we need only look at one of the possible solutions.

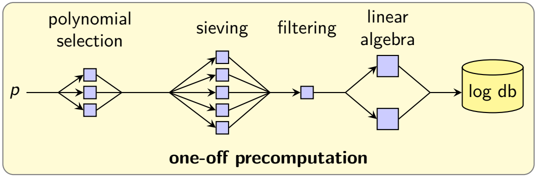
This brings only benefits, since in practice:

- r is at most $(\deg f_0 - 1) + (\deg f_1 - 1) \leq 5$. (DLP240: $r = 4$).
- We like to choose n so that some distribution is possible. (DLP240: $n = 16$).

Plan

Where are we?

After the linear algebra step, we have a big database of (virtual) logarithms for many ideals and Schirokauer maps.



Cost thus far:

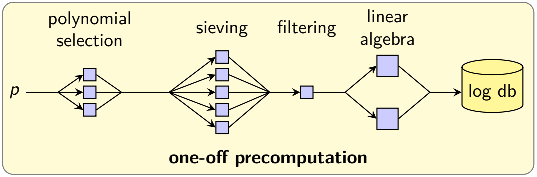
- the analysis is exactly the same.
- Joux-Lercier polynomial selection doesn't change anything.

Cost of NFS-DL precomputation

$$\text{relation collection} \approx \text{sieving} \approx L_p(1/3, (64/9)^{1/3} + o(1)).$$

Where are we?

After the linear algebra step, we have a big database of (virtual) logarithms for many ideals and Schirokauer maps.



This is not enough.

- We want to compute $\mathbf{L}(x)$ for arbitrary things.
- Anyway, the keys of our database are NOT elements of $\mathbb{F}_p^\times$.

How do we compute $\mathbf{L}(h)$ for an arbitrary $h \in \mathbb{F}_p^\times$, given our database?

Not so easy to get it right

General idea

If $h \in \mathbb{F}_p^\times$ is something that decomposes along our factor base, we're good.
Caveat: our factor base consists of ideals.

The process consists of two steps.

- An initial lift of our target data on one of the sides.
- A descent process that allows us to relate our target to the things that are referenced in the database.

The earliest version of this process, although in a different context, appeared in Coppersmith's 1984 paper on $L(1/3)$ discrete logarithms in $\mathbb{F}_{2^n}$.

Random self-reduction

If we want to compute $\mathbf{L}(h)$, note that for any invertible e mod ℓ, we have:

$$\mathbf{L}(h) = \frac{1}{e}\mathbf{L}(h^e).$$

This implies that we have worst-case to average-case reduction for this problem.

We will use this randomization property soon.

Lifting / Finding preimages

Our element $h \in \mathbb{F}_p^\times$ must be lifted on one of the two sides.

- From a mathematical standpoint, both sides would “work”.
- In practice, one side is often “smaller” than the other, and therefore preferred.

In the Joux-Lercier case, we choose the larger-degree, smaller-coefficients polynomial f_0, whose degree is asymptotically

$$\deg f_0 \approx \log_{\log p} L_p(1/3, \cdot).$$

In what follows, $(f, K, \alpha, \ldots)$ are shorthands for $(f_0, K_0, \alpha_0, \ldots)$

Lifting / Finding preimages

The map from K to $\mathbb{F}_p^\times$ sends α to m mod p.

Naive approach

First stategy: lift h to some integer representative, viewed as an element of K.

Our hope: maybe h (or h^e) is a smooth integer.
Would it be useful?

Can we use a smooth integer?

We are able to evaluate **L** on any element which factors into ideals of norm below $B \approx L_p(1/3)$.

We have:

- $h \approx L_p(1)$. In time $L_p(1/3)$, we can hope for some h^e to be $L_p(2/3)$-smooth.
- This would be useful if we were lifting to the rational field, but not here.
  The norm $\mathrm{Norm}_{K/\mathbb{Q}}$ is a lot larger! Not all small primes factor into small prime ideals in the number field.
- We have $\mathrm{Norm}_{K/\mathbb{Q}}(h) \approx L_p(1)^{\log_{\log p} L_p(1/3)} \approx L_p(4/3)$, which is way too large.

Rational reconstruction

Using rational reconstruction

Second approach: lift h to a rational representative such that

$$h \equiv \frac{u}{v} \mod p.$$

Such (u, v) can be obtained with the extended Euclidean algorithm, with $u \approx v \approx \sqrt{p}$.

Two half-size numbers have better chances of smoothness than one that is twice as large.

Yet, this still does not work, as it does not change the asymptotics.

Use sieving

Using rational reconstruction + sieving

Third approach: use two consecutive rational representations of h:

$$h \equiv \frac{u_0}{v_0} \equiv \frac{u_1}{v_1} \mod p.$$

Then, look for multipliers k_0 and k_1 such that

- $k_0 u_0 + k_1 u_1$ is a smooth integer;
- $k_0 v_0 + k_1 v_1$ is a smooth integer.

This can be done with sieving

This might have a tremendous practical impact, but does not really solve the problem. We're dealing with BIG integers and norms here.

Lift to algebraic numbers

Use the fact that $\alpha \to m$ to lift h to something that involves α.

We have more freedom, we have some hope to obtain smaller things, and in particular a smaller algebraic norm.

Lifting to algebraic numbers

Joux–Lercier polynomial selection:
two polynomials of degree $(d, d+1)$

Baby example for a 30-digit prime factor.

$$\begin{aligned} p &= 303754470965461741563252151428 7 \\ f &= x^3 - x + 1 \\ m &= 178827864877625171826943706505 7 \\ g &= 5383086967x^2 - 13169419660x - 2588305959 \end{aligned}$$

$\rightarrow$ no rational side.

Lifting to algebraic numbers (Joux-Lercier)

Lift $h \in \mathbb{F}_p^\times$ to a fraction $\dfrac{u_0 + \cdots + u_d\alpha^d}{v_0 + \cdots + v_d\alpha^d} \mapsto h$.

Define the following lattice

$$L = \begin{bmatrix} p & & & & & & \\ -m & 1 & & & & & \\ & \ddots & \ddots & & & & \\ & & -m & 1 & & & \\ h & 0 & \dots & 0 & 1 & & \\ & \ddots & & & & \ddots & \\ & & h & & & & 1 \end{bmatrix}$$

Each row vector in this lattice is such that

$$(u_0 + \cdots + u_d\alpha^d) - h(v_0 + \cdots + v_d\alpha^d) \mapsto 0.$$

Lifting to algebraic numbers (Joux-Lercier)

target: $h = 92800609832959449330691138186 \in \mathbb{F}_p^\times$.

$$\text{Matrix: } \begin{bmatrix} 303754470965461741563252151428 & 0 & 0 & 0 & 0 & 0 \\ -178827864877625171826943706505 & 1 & 0 & 0 & 0 & 0 \\ 0 & -178827864877625171826943706505 & 1 & 0 & 0 & 0 \\ 92800609832959449330691138186 & 0 & 0 & 1 & 0 & 0 \\ 0 & 92800609832959449330691138186 & 0 & 0 & 1 & 0 \\ 0 & 0 & 92800609832959449330691138186 & 0 & 0 & 1 \end{bmatrix}$$

Apply LLL:

$$\begin{bmatrix} 52606 & 44203 & -25671 & -6448 & -66975 & 5015 \\ 25671 & 26935 & 44203 & -5015 & -1433 & -66975 \\ 44203 & -69874 & -26935 & -66975 & 71990 & 1433 \\ 71307 & -42105 & 21380 & 106583 & -10333 & 35519 \\ -49927 & -50582 & 20725 & -71064 & -131769 & -25186 \\ 3531 & -5555 & -115510 & 101265 & 36952 & -39608 \end{bmatrix}$$

Lifting to algebraic numbers (Joux-Lercier)

target: $h = 92800609832959449330691138186 \in \mathbb{F}_p^\times$, lift as $y \in \mathbb{Z}$
Each row of LLL output is $[u_0, u_1, u_2, v_0, v_1, v_2]$:

$$\texttt{PREIMAGE}(h) = \frac{u_0 + u_1 x + u_2 x^2}{v_0 + v_1 x + v_2 x^2}$$

First row gives:

$$h \equiv \frac{52606 + 44203m - 25671m^2}{-6448 - 66975m + 5015m^2} \bmod p$$

Is this a win?

This method encompasses rational reconstruction, and is compatible with sieving as well.

Is the norm smaller?

Some analysis

- The dimension of the lattice is $2d+2 \approx \log_{\log p}(L_p(1/3))$.
- Its determinant is $p = L_p(1)$.
- The L^2 norm of its small vectors are expected to be

  $$\text{(approximation factor)} \times \det^{1/\dim} \approx L_p(1)^{1/\log_{\log p} L_p(1/3)} \approx L_p(2/3).$$

- The algebraic norm of the resulting elements can be expected to be

  $$L_p(2/3)^d \approx L_p(1).$$

It is a win, because the algebraic norm is what we need for the rest of the process.

Summary of the initial step

- Try many h^e simultaneously.
- Form lattices, pick short vectors. Perhaps do some sieving.
- Compute algebraic norms, try to factor them into prime factors below some bound $B_{\text{init}} \approx L_p(2/3)$. ECM is here to help.
- This costs $L_p(1/3)$.

We are left with some algebraic number whose ideal factorization involves only prime ideal of norm below $L_p(2/3)$.

An example

The initial step for the DLP-240 individual logarithm computation obtains $\frac{u(\alpha)}{v(\alpha)} \mapsto (\text{target})^e$, with:

$$\begin{aligned}
U(\alpha) = {} & 1111581465498762143665172631 3 \\
& +84293370936324477859384006099\alpha \\
& +101013631005385213436119240940\alpha^2 \\
& -4019106371406544983054585709 67\alpha^3 \\
V(\alpha) = {} & -14022053826479031727483421649 3 \\
& +153214328933512769640372578392\alpha \\
& -104892106869531614533623596 57\alpha^2 \\
& -624779326557421402049330473855\alpha^3 \\
U(\alpha)\mathcal{O}_K = {} & \mathfrak{p}_3 \times \mathfrak{p}_5 \times \mathfrak{p}_{113} \times \mathfrak{p}_{1543} \times \mathfrak{p}_{6dd} \times \mathfrak{p}_{10dd} \times \mathfrak{p}_{18dd} \times \mathfrak{p}_{20dd} \times \mathfrak{p}_{20dd} \times \mathfrak{p}_{22dd} \times \mathfrak{p}_{25dd} \\
V(\alpha)\mathcal{O}_K = {} & \mathfrak{p}_3 \times \mathfrak{p}_7 \times \mathfrak{p}_{41} \times \mathfrak{p}_{113} \times \mathfrak{p}_{401} \times \mathfrak{p}_{14dd} \times \mathfrak{p}_{15dd} \times \mathfrak{p}_{19dd} \times \mathfrak{p}_{22dd} \times \mathfrak{p}_{24dd} \times \mathfrak{p}_{26dd}
\end{aligned}$$

Plan

The descent

The descent is the process through which we will try to compensate all the oversize ideals in the previous factorization.

Note

The initialization step used only one side (side 0) of the NFS diagram.
The descent process involves BOTH SIDES.

The main idea is to find some $a - bx$ such that:

- $(a - b\alpha_0)\mathcal{O}_{K_0}$ is divisible by one of these oversize ideals.
- Both $(a - b\alpha_0)\mathcal{O}_{K_0}$ AND $(a - b\alpha_1)\mathcal{O}_{K_1}$ are somewhat smooth ideals.

It may be too much of a stretch to aim at our ultimate factor base bound just yet, though.

The descent

Starting point (initialization):

- $\left\{\begin{array}{l}\text{side 0: } S_0(\alpha_0) = \text{something},\\ \text{side 1: } S_1(\alpha_1) = 1,\end{array}\right\}$ and $\frac{S_0(m)}{S_1(m)} = \text{target}$.
- $S_0(\alpha_0)\mathcal{O}_{K_0}$ is divisible by some large ideals.

With a properly chosen $a - bx$, we have after this first step:

- $\left\{\begin{array}{l}S_0'(\alpha_0) = S_0(\alpha_0)/(a - b\alpha_0),\\ S_1'(\alpha_1) = S_1(\alpha_1)/(a - b\alpha_1),\end{array}\right\}$ and $\frac{S_0'(m)}{S_1'(m)} = \text{target}$.
- $S_0'(\alpha_0)\mathcal{O}_{K_0}$ has one large ideal less, and might have gained a few other ideals.
- $S_1'(\alpha_1)\mathcal{O}_{K_1}$ might have gained a few other ideals.

Finding the multiplier

How do we choose $a - bx$? With special-q sieving!

- The ideal that we're trying to cancel has a norm $\approx B_{\text{init}}$.
- If we can obtain smoothness with a new bound $B_1 \leq B_{\text{init}}^c$ for some $c < 1$, we're probably good.

Analysis of the descent steps

Here's how the analysis of the descent could work.

$$\begin{aligned}
\text{Norm}\,\mathfrak{q} &\approx B_{\text{init}} \approx L_p(2/3, k)\\
\mathfrak{q}\text{-lattice coefficients} &\approx \sqrt{\text{Norm}\,\mathfrak{q}} \approx L_p(2/3, k/2)\\
\text{Norm}(a - b\alpha_0) &\approx L_p(1, \text{something} \times k/2)\\
B_1 &\approx L_p(2/3, kc).\\
\text{prob. smoothness} &\approx L_p(1/3, -\frac{1}{6c} \times \text{something})
\end{aligned}$$

We might keep going, and reach $L_p(2/3, kc^n)$ after n steps.

Messy, but works. This yields a cost of $L(1/3, \cdot)$, smaller than the rest.

This is pretty much what we do in practice, although the different bounds are rather chosen by hand.

Alternative descent strategy

Another approach could be not to search for $a - b\alpha$, but for larger degree polynomials with degree $\delta = \log_{\log p} L_p(1/6, \cdot)$ (at the first step).

- The $\mathfrak{q}$-lattice coefficients are smaller in this case: $(\operatorname{Norm} \mathfrak{q})^{\frac{1}{\delta}}$.
- This makes it possible to obtain smaller norms.

Alternative descent strategy

$$\text{Norm } u \approx L(1)$$

$$\text{Norm } \mathfrak{q} \approx L(1 - 1/3 = 2/3 = 1/3 + 1/3)$$

$$\rightarrow \delta = \dim(\mathfrak{q}\text{-lattice}) \approx \log_{\log p} L_p(1/6)$$

$$\rightarrow \|\textstyle\sum_i a_i \alpha^i = \phi(\alpha)\|_\infty \approx |a_i| \approx L_p(2/3 - 1/6 = 1/2 = 1/3 + 1/6)$$

$$\rightarrow \text{Norm}(\phi(\alpha)) \approx \|\phi\|_\infty^d \|f\|^\delta \approx L_p(\max(1/3 + 1/6 + 1/3, 2/3 + 1/6) = 5/6)$$

$$\text{Norm } \mathfrak{q}' \approx L(5/6 - 1/3 = 1/2 = 1/3 + 1/6)$$

$$\rightarrow \delta = \dim(\mathfrak{q}\text{-lattice}) \approx \log_{\log p} L_p(1/12)$$

$$\rightarrow \|\textstyle\sum_i a_i \alpha^i = \phi(\alpha)\|_\infty \approx |a_i| \approx L_p(1/2 - 1/12 = 5/12 = 1/3 + 1/12)$$

$$\rightarrow \text{Norm}(\phi(\alpha)) \approx \|\phi\|_\infty^d \|f\|^\delta \approx L_p(\max(1/3 + 1/12 + 1/3, 2/3 + 1/12) = 9/12)$$

etc. . .

Eventually (and probably very quickly!) $\log_{\log p} L_p(1/(3 \cdot 2^i))$ is very small. For comparison, $\log_{\log p} L_p(1/6) \approx \sqrt{d}$.

Alternative descent strategy

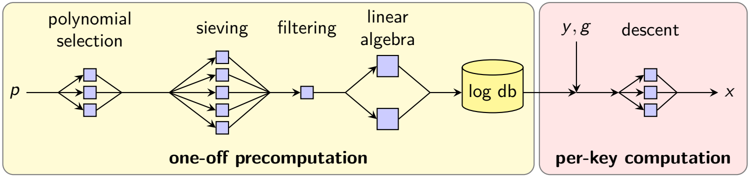
Asymptotically, this gives a much nicer descent complexity, quite in line with Coppersmith's descent idea from 1984.

In practice, higher degree sieving is not used in the descent, but it was a close call for the first step of some of the latest computations.

Overall complexity

While it is a complicated process, the descent is comparatively a lot cheaper than the rest of the computation.

- The initialization step is the most expensive part, with cost $L_p(1/3, 1.232 + o(1))$ with an early-abort strategy.
- The rest of the descent is cheaper.



Typical data (elapsed time using many machines):

	precomputation	per-key
Logjam (512 bits)	week	minutes
DLP-240 (795 bits)	months	hours