Discrete Logarithm Cryptanalyses:

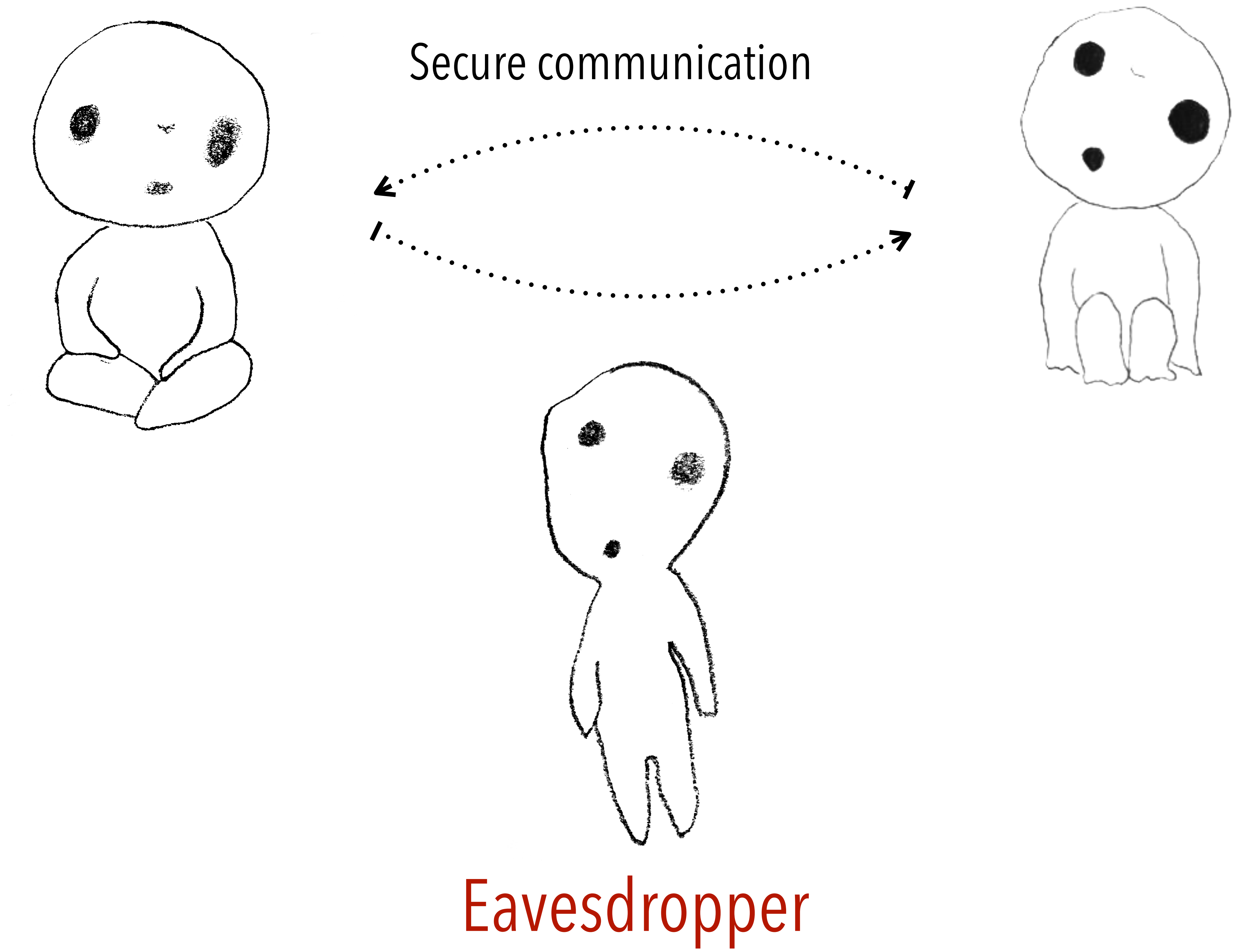
Number Field Sieve and Lattice Tools for Side-Channel Attacks

Thesis defense

Gabrielle De Micheli, 25th of May 2021

Under the supervision of Pierrick Gaudry and Cécile Pierrot

What? Why? Where? Cryptography …



Cryptographic protocols for:

- Confidentiality (encryption schemes)
- Authentication and non-repudiation (signature schemes)
- Integrity and validity of data (hash functions)
- …

Hard problems for Cryptography

Use (hopefully) intractable problems to construct cryptographic primitives.

Start from…

- factorisation
- discrete logarithm
- lattice problems
- isogeny problems
- …

… to obtain:

- encryption schemes
- signature schemes
- hash functions
- …

Hard problems for Cryptography

Use (hopefully) intractable problems to construct cryptographic primitives.

Start from…

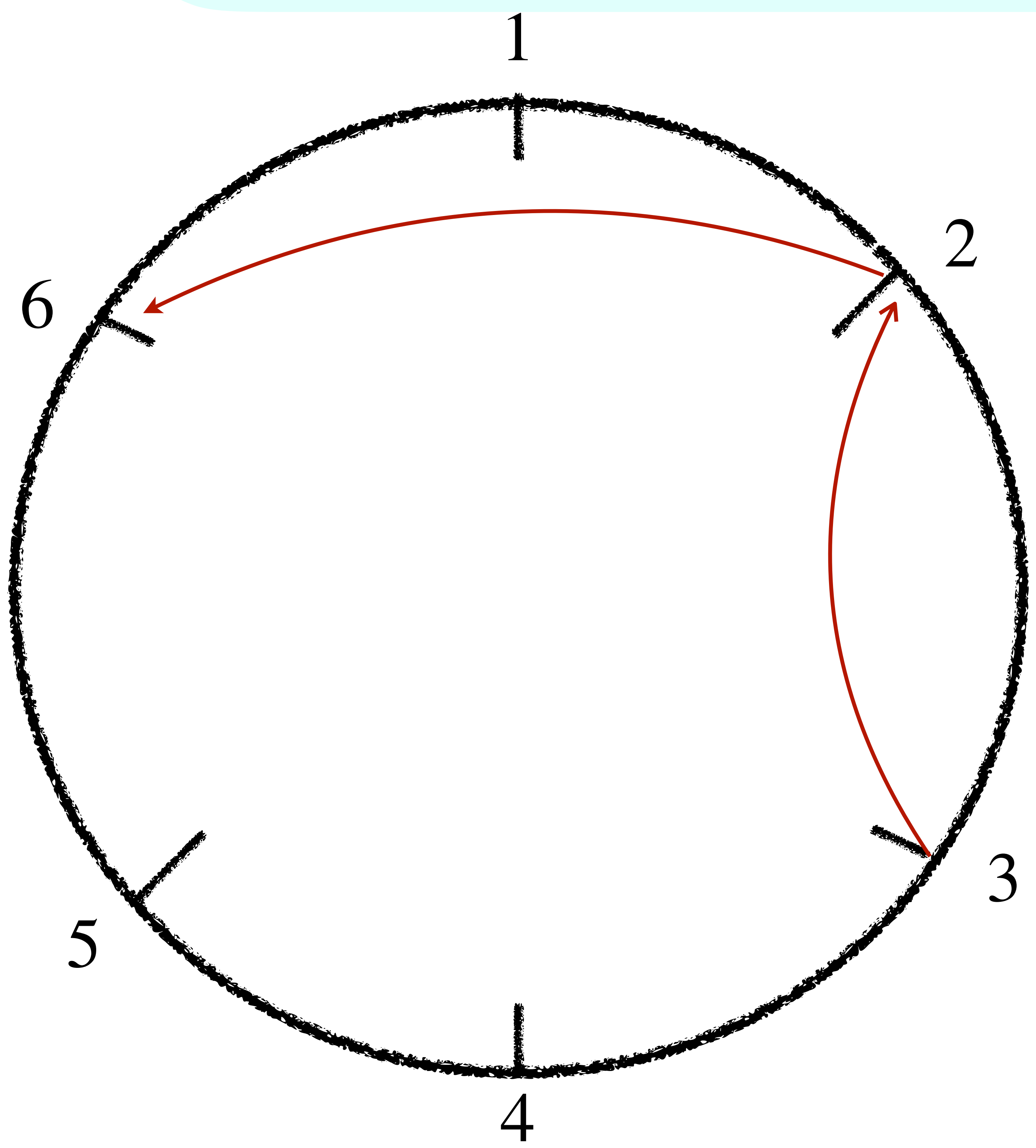
- factorisation
- discrete logarithm
- lattice problems
- isogeny problems
- …

… to obtain:

- encryption schemes
- signature schemes
- hash functions
- …

What is a discrete logarithm?

Definition: Given a finite cyclic group G of order n, a generator $g \in G$ and some element $h \in G$, the discrete logarithm of h in base g is the element $x \in [0,n)$ such that $g^x = h$.



Example: $G = \mathbb{Z}_7^\times,\ g = 3,$

$h = 6 \in \mathbb{Z}_7^\times,$

$$g^1 \equiv 3 \pmod 7$$
$$g^2 = 9 \equiv 2 \pmod 7$$
$$g^3 = 27 \equiv 6 \pmod 7$$

The discrete logarithm of h in base g is 3.

The discrete logarithm problem (DLP)

Definition: Given a finite cyclic group G of order n, a generator $g \in G$ and some element $h \in G$, **find** the element $x \in [0,n)$ such that $g^x = h$.

Computing the inverse, a modular exponentiation is easy:

$$g^x = \underbrace{g \cdot g \cdot \cdots \cdot g}_{x}$$

algorithms in $O(\log(x))$

Solving DLP can be hard (depending on the group G):

$$h = \underbrace{g \cdot g \cdot \cdots \cdot g}_{??}$$

Motivation: why do we care about modular exponentiation?

Many protocols use modular exponentiation where the exponent is a secret.

Example 1: Diffie-Hellman key exchange [DH76]

- Public data: $g, g^a, g^b \in G$
- Shared key: $g^{ab} \in G$

Ephemeral Diffie Hellman

Technical Details
Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)

Example 2: pairing-based protocols

- Identity-based encryption/signature schemes [BF01], [CC03]
- Short signature schemes (eg, BLS signatures [BLS01])

Security based on assumptions that become false if DLP is broken.

[DH76]: W. Diffie, M. Hellman, New directions in cryptography. Trans. Info. Theory, 1976
[BF01]: D. Boneh, M. Franklin, Identity-based encryption from Weil pairing. Crypto'01
[CC03]: J. Cha, J. Cheon, An identity-based signature from gap Diffie-Hellman groups. PKC'03
[BLS01]: D. Boneh, B. Lynn, H. Shacham, Short signatures from the Weil pairing. Asiacrypt'01

In my work

How can we assess the security of protocols in which a modular exponentiation involving a secret exponent is performed?

- Estimate the hardness of DLP in the groups considered by the protocols.
- Look at implementation vulnerabilities during fast exponentiation.

An example: EPID protocol in Intel SGX

- What is EPID? a protocol to allow remote attestation of a hardware platform without compromising the device's identity.

- The protocol includes a signing algorithm that uses pairings.

  - secret key includes the element $f \in_R \mathbb{Z}_q$

- How can we recover f?

  - During the protocol, consider a random secret nonce $r \in \mathbb{Z}_q$

  - Compute an exponentiation X^r

  - Outputs the element $s \leftarrow r + cf$ ($c =$ hash of known values)

How can we recover the secret f ?

Since $s \leftarrow r + cf$, if we recover r, we directly get f.

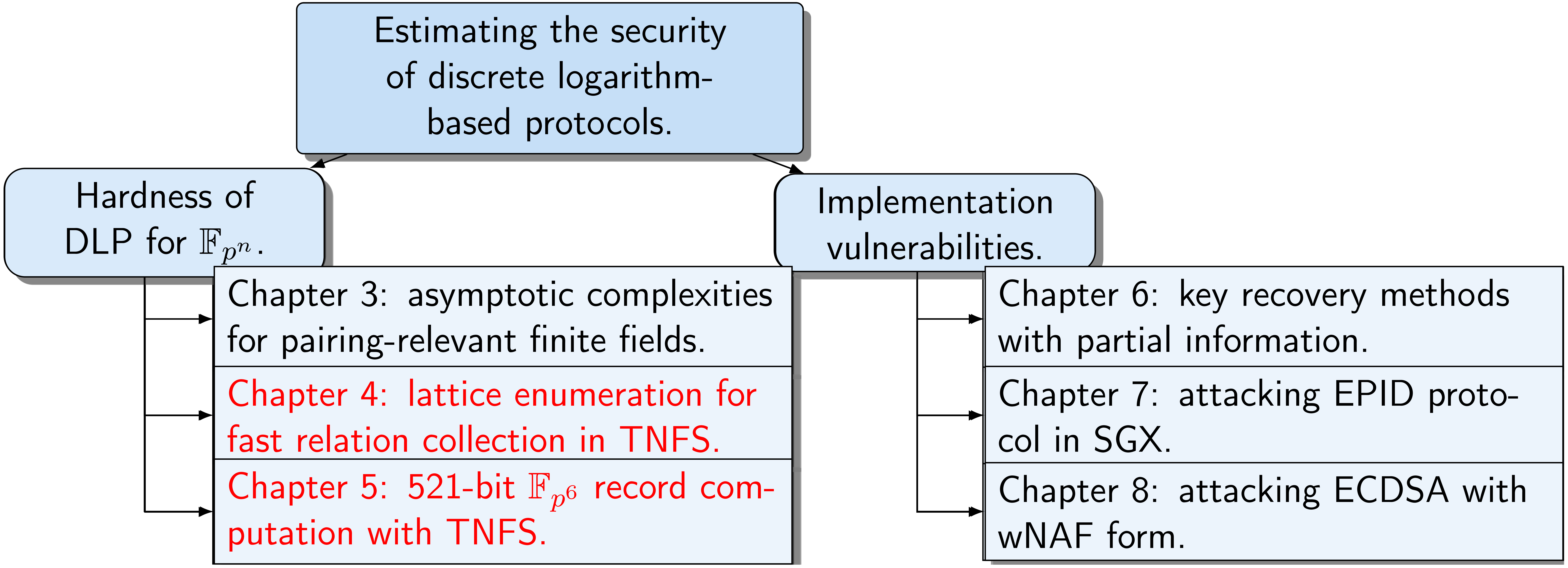
The protocol uses a 256-bit elliptic curve Fp256BN (embedding degree 12).

If we have as target X^r :

1. Solve DLP to find exponent r in 3072-bit finite field $\mathbb{F}_{p^{12}}$.

2. Look at implementation vulnerabilities during the computation of X^r.

Thesis contributions



Implementation vulnerabilities

Exploiting leakage from side-channels

How can we recover the secret f in EPID ?

Since $s \leftarrow r + cf$, if we recover r, we directly get f.

The protocol uses a 256-bit elliptic curve Fp256BN (embedding degree 12).

If we have as target X^r :

1. ~~Solve DLP to find exponent~~ $\cancel{r}$ ~~in 3072-bit finite field~~ $\cancel{\mathbb{F}_{p^{12}}}$ ~~.~~

2. Look at implementation vulnerabilities during the computation of X^r.

Recovering partial information on r is enough to obtain f.

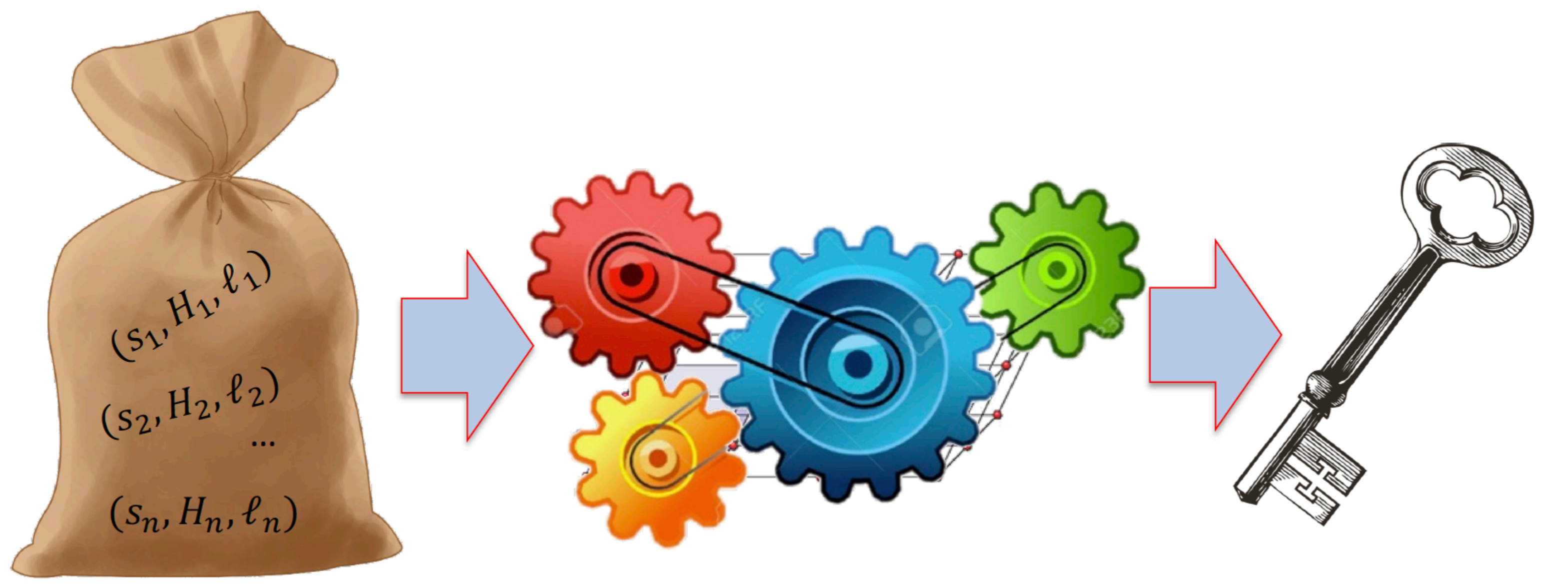
What is partial information and where does it come from?

1. Side-channel attacks, in particular cache attacks.

2. Many microarchitectural side-channel attacks use variations in execution time as the source of leakage.

3. Fast modular exponentiation algorithm are rarely constant-time !

In this thesis, we focus on how to what to do with the leaked information.



Key recovery methods

I have obtained the following type of incomplete information about the secret key. Does it allow me to efficiently recover the rest of the key?

Methods depend on:

- algorithm considered
- nature of the information leaked

Scheme	Secret information	Bits known	Technique
RSA	$p \geq$ 50% most significant bits		Coppersmith's method
RSA	$p \geq$ 50% least significant bits		Coppersmith's method
RSA	p middle bits		Multivariate Coppersmith
RSA	p multiple chunks of bits		Multivariate Coppersmith
RSA	$> \log\log N$ chunks of p		Open problem
RSA	$d \pmod{p-1}$ MSBs		Coppersmith's method
RSA	$d \pmod{p-1}$ LSBs		Coppersmith's method
RSA	$d \pmod{p-1}$ middle bits		Multivariate Coppersmith
RSA	$d \pmod{p-1}$ chunks of bits		Multivariate Coppersmith
RSA	d most significant bits		Not possible
RSA	$d \geq$ 25% least significant bits		Coppersmith's method
RSA	$\geq$ 50% random bits of p and q		Branch and prune
RSA	$\geq$ 50% of bits of $d \pmod{p-1}$ and $d \pmod{q-1}$		Branch and prune
(EC)DSA	MSBs of signature nonces		Hidden Number Problem
(EC)DSA	LSBs of signature nonces		Hidden Number Problem
(EC)DSA	Middle bits of signature nonces		Hidden Number Problem
(EC)DSA	Chunks of bits of signature nonces		Extended HNP
EC(DSA)	Many bits of nonce		Scales poorly
Diffie-Hellman	Most significant bits of shared secret g^{ab}		Hidden Number Problem
Diffie-Hellman	Secret exponent a		Pollard kangaroo method
Diffie-Hellman	Chunks of bits of secret exponent		Open problem

Recovering cryptographic keys from partial information, by example, with Nadia Heninger, **Eprint 2020/1507**

The (Extended) Hidden Number Problem

In my work:

- Lattice-based approach
- Optimizing the lattice construction
- Error handling
- Concrete attacks:

(EC)DSA	MSBs of signature nonces	Hidden Number Problem
(EC)DSA	LSBs of signature nonces	Hidden Number Problem
(EC)DSA	Middle bits of signature nonces	Hidden Number Problem
(EC)DSA	Chunks of bits of signature nonces	Extended HNP
EC(DSA)	Many bits of nonce	Scales poorly
Diffie-Hellman	Most significant bits of shared secret g^{ab}	Hidden Number Problem

- EPID signing algorithm — 37 signatures with HNP to recover the key in 4.5 seconds

- ECDSA with wNAF — 3 signatures with EHNP to recover the key in 5 days

CacheQuote: Efficiently Recovering Long-term Secrets of SGX EPID via Cache attacks, with Fergus Dall, Thomas Eisenbarth, Daniel Genkin, Nadia Heninger, Ahmad Moghimi and Yuval Yarom, at **CHES 2018**

A Tale of Three Signatures: practical attack of ECDSA with wNAF, with Cécile Pierrot and Rémi Piau, at **Africacrypt 2020**

CVE: Common Vulnerabilities and Exposures

Attacking EPID signing algorithm

37 signatures with HNP to recover the key in 4.5 seconds

CVE-2018-3691 Detail

Current Description

Some implementations in Intel Integrated Performance Primitives Cryptography Library before version 2018 U3.1 do not properly ensure constant execution time.

— Hide Analysis Description

Analysis Description

Some implementations in Intel Integrated Performance Primitives Cryptography Library before version 2018 U3.1 do not properly ensure constant execution time.

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** **4.7 MEDIUM** **Vector:** CVSS:3.0/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:N/A:N

Attacking the primitive

Hardness of DLP for $\mathbb{F}_{p^n}$

The discrete logarithm problem over finite fields

Definition: Given a finite cyclic group G of order n, a generator $g \in G$ and some element $h \in G$, find the element $x \in [0,n)$ such that $g^x = h$.

What group G should be considered?

$$\cancel{(\mathbb{Z}/n\mathbb{Z}, +)}$$

- Prime finite fields $\mathbb{F}_p^\times$
- Finite fields $\mathbb{F}_{p^n}^\times$
- Elliptic curves over finite fields $\mathscr{E}(\mathbb{F}_p)$
- Genus 2 hyperelliptic curves

The discrete logarithm problem over finite fields

Definition: Given a finite cyclic group G of order n, a generator $g \in G$ and some element $h \in G$, find the element $x \in [0,n)$ such that $g^x = h$.

What group G should be considered?

$$\cancel{(\mathbb{Z}/n\mathbb{Z}, +)}$$

- Prime finite fields $\mathbb{F}_p^\times$
- Finite fields $\mathbb{F}_{p^n}^\times$
- Elliptic curves over finite fields $\mathscr{E}(\mathbb{F}_p)$
- Genus 2 hyperelliptic curves

Evaluating the hardness of DLP over $\mathbb{F}_{p^n}$

- Many different algorithms to solve DLP in $\mathbb{F}_{p^n}$.
- Their complexities depend on the relation between the characteristic p and the extension degree n.

A useful notation: the L-notation $\boxed{L_{p^n}(\alpha, c) = \mathbf{exp}((c + o(1))\log(p^n)^{\alpha}\log\log(p^n)^{1-\alpha})}$

for $0 \leqslant \alpha \leqslant 1$ and $c > 0.$

For complexities:

- When $\alpha \to 0$: $\exp(\log\log p^n) \approx \log p^n$, polynomial-time
- When $\alpha \to 1$: p^n, exponential-time

In the <u>middle</u>: subexponential-time

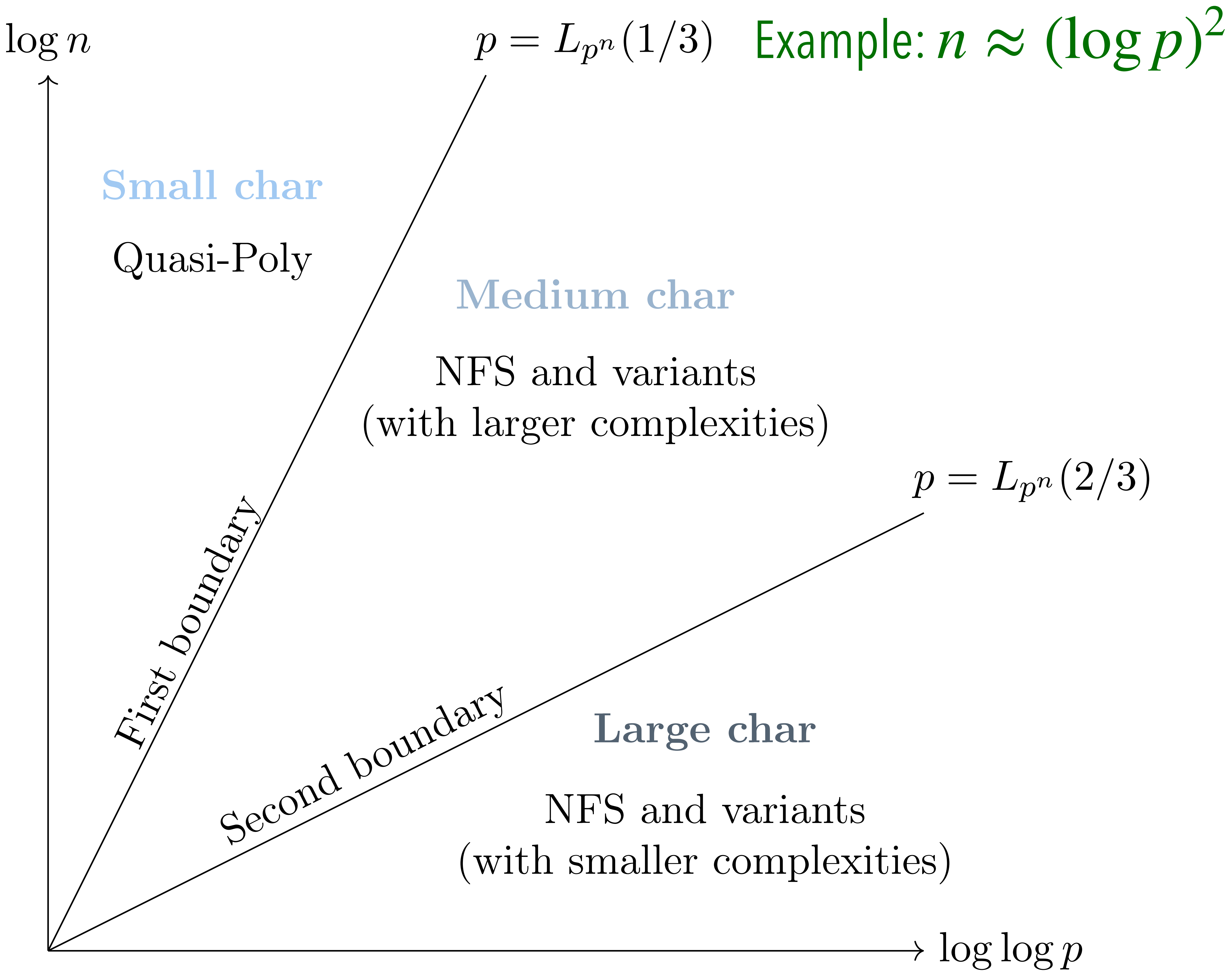
Three families of finite fields

Finite field $\mathbb{F}_{p^n}$ with $p = L_{p^n}(\alpha, c)$

- Different algorithms are used in the different areas.

- Algorithms don't have the same complexity in each area.

What are these algorithms?



They all come from a family known as index calculus algorithms.

Index calculus algorithms

Consider a finite field $\mathbb{F}_{p^n}$

Factor basis: $\mathscr{F} =$ small set of small elements

Three main steps:

- Relation collection: find relations between the elements of $\mathscr{F}$.
- Linear algebra: solve a system of linear equations where the unknowns are the discrete logarithms of the elements of $\mathscr{F}$.
- Individual logarithm/Descent: for a target element $h \in \mathbb{F}_{p^n}^{\times}$, compute the discrete logarithm of h.

A lot of algorithms

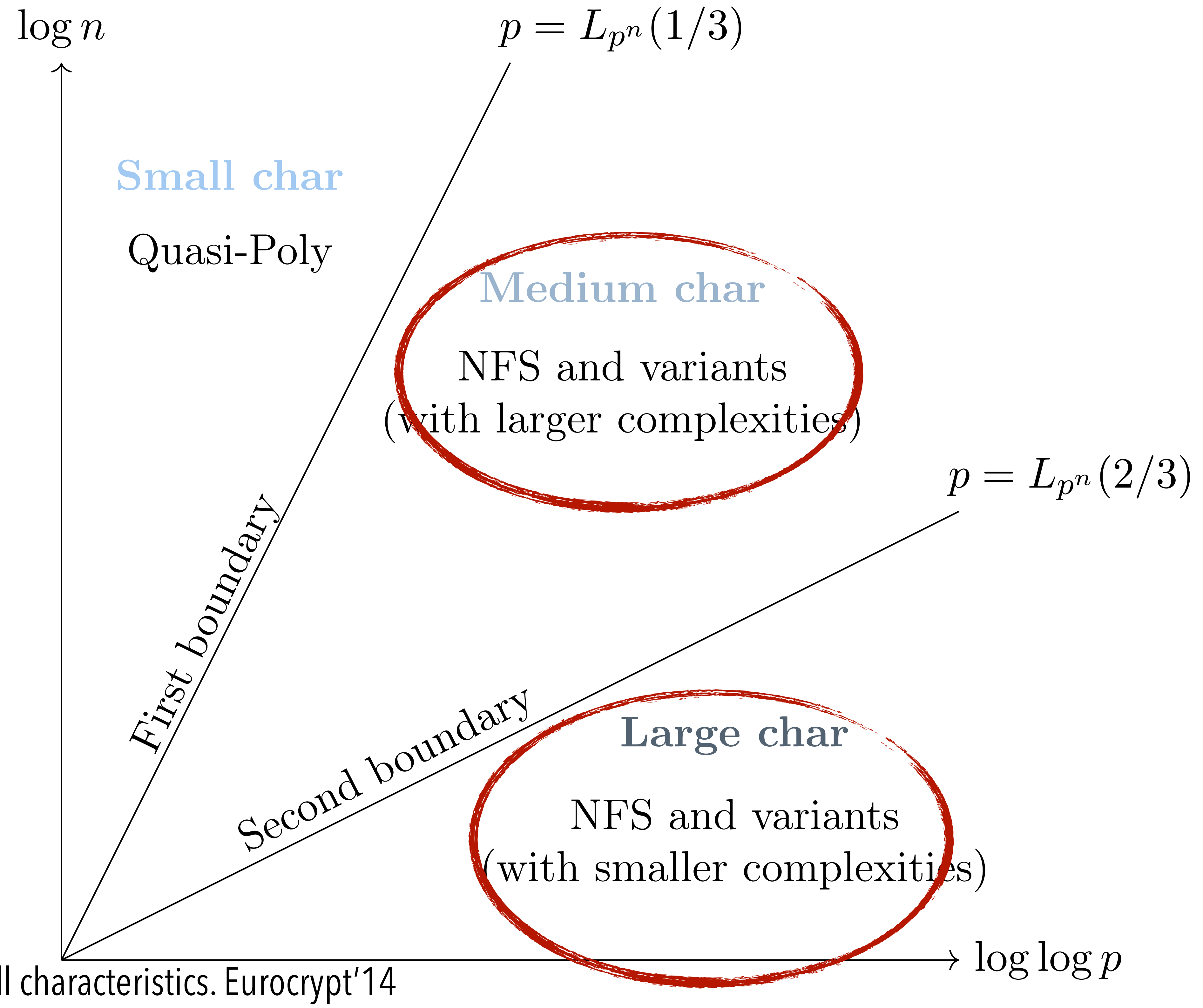
- Small characteristics: Quasi-Polynomial algorithms [BGJT14, KW19] (with only a descent step) and Function Field Sieve [Adl94]

- Medium and large characteristics: Number Field Sieve (NFS) [Gor93] and its variants

We focus on <u>medium and large characteristic</u> finite fields.

Why?

Finite fields used in practice for example $\mathbb{F}_{p^6}$ for MNT-6 elliptic curves in zk-SNARKS.



[Adl94]: L. Adleman, The Function Field Sieve. ANTS'94

[Gor98]: D. Gordon, Discrete Logarithms in GF(P) Using the Number Field Sieve. Journal on Discrete Mathematics'93

[BGJT14]: R. Barbulescu, P. Gaudry, A. Joux, E. Thomé, A heuristic quasi-polynomial time algorithm for discrete logarithm in finite fields of small characteristics. Eurocrypt'14

[KW19]: T. Kleinjung, B. Wesolowski, Discrete logarithms in quasi-polynomial time in finite fields of fixed characteristic. 2019

Back to the hardness of DLP on $\mathbb{F}_{p^n}$

Two ways of evaluating the hardness of DLP:

1. Study the complexities of these algorithms.

$$L_{p^n}(1/3, c)$$

Specificity	Algorithm	Medium characteristic	2nd boundary	Large characteristic
None	NFS	96	48	64
	MNFS	89.45	45.00	61.93
	TNFS	–	–	64
	MTNFS	–	–	61.93
Composite n	exTNFS	48	–	–
	MexTNFS	45.00	–	–
Special p	SNFS	$64\left(\frac{\lambda+1}{\lambda}\right)$	$\star$	32
	STNFS	–	–	32
Composite n and special p	SexTNFS	32	$\star$	32

2. Perform record computations.

$\mathbb{F}_{2^{30750}}$	30750	QP	2019 [GKL+20]
$\mathbb{F}_{3^{3054}}$	4841	QP	2016 [ACMM+18]

$\mathbb{F}_{p^{50}}$	1051	FFS	2020 [MSST20]

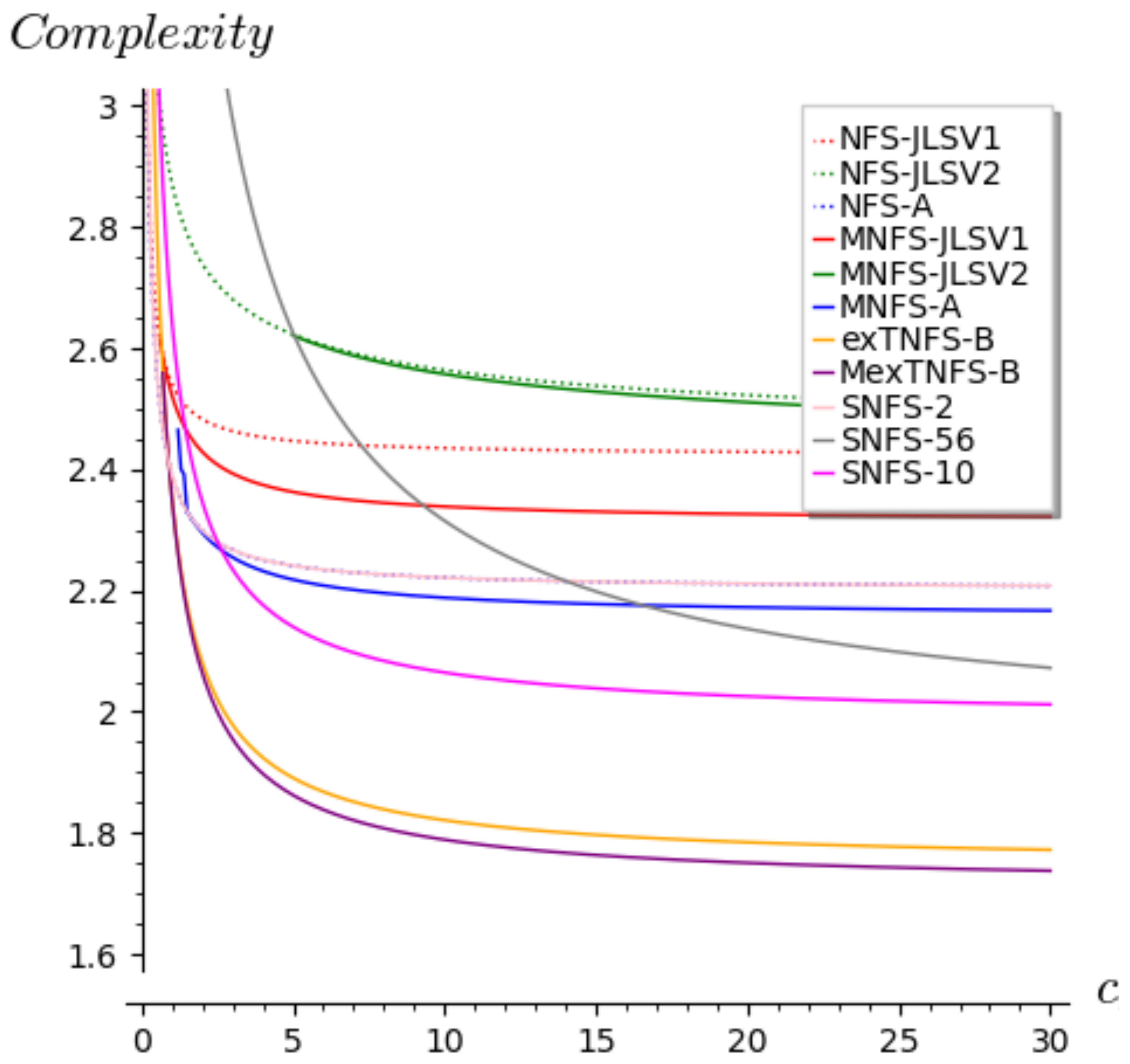
$\mathbb{F}_{p^6}$	521	TNFS	2021 this thesis
$\mathbb{F}_{p^6}$	423	NFS-HD	2020 [MR21]
$\mathbb{F}_{p}$	795	NFS	2020 [BGG+20]
$\mathbb{F}_{p^6}$	422	NFS-HD	2016 [GGMT17]
$\mathbb{F}_{p^5}$	324	NFS-HD	2016 [GGM17]
$\mathbb{F}_{p}$	1024	SNFS	2016 [FGHT17]
$\mathbb{F}_{p^3}$	593	NFS	2016 [Gré16]
$\mathbb{F}_{p}$	768	NFS	2016 [KDL+17]
$\mathbb{F}_{p^3}$	508	NFS	2016 [GMT16]

$0 \qquad \frac{1}{3} \qquad \frac{2}{3} \qquad 1 \qquad \alpha$

Small char | Medium char | Large char

In this thesis

1. We studied the asymptotic complexity of all these variants at the first boundary case: $p = L_{p^n}(1/3, c)$.



One conclusion from this work: **estimates for 128-bit security and asymptotic analysis do not match.**

2. We ran large-scale experiments with the variant TNFS.

Asymptotic complexities of discrete logarithm algorithms in pairing-relevant finite fields, with Pierrick Gaudry and Cécile Pierrot, at **Crypto 2020**

https://members.loria.fr/AGuillevic/pairing-friendly-curves/#pairing-friendly-curves-at-the-128-bit-security-level

Why do we do record computations?

It is important to choose the right key size.

- Too large: needlessly expensive computations
- Too small: insecure

Agency	Date	Size of group	Size of key
NIST	2019-2030	2048	224
	> 2030	3072	256
ANSSI	2021-2030	2048	200
	> 2030	3072	200

Running-time of discrete logarithm algorithms is hard to predict.

Record computations provide information for assessing key lifetime.

A first record computation with exTNFS [KB16] $\mathbb{F}_{p^n} = \mathbb{F}_{p^{\eta\kappa}} = \mathbb{F}_{P^\kappa}$

- Why did we choose exTNFS?

$n = \eta\kappa$

Specificity	Algorithm	Medium characteristic	2nd boundary	Large characteristic
None	NFS	96	48	64
	MNFS	89.45	45.00	61.93
	TNFS	–	–	64
	MTNFS	–	–	61.93
Composite n	exTNFS	48	–	–
	MexTNFS	45.00	–	–
Special p	SNFS	$64\left(\frac{\lambda+1}{\lambda}\right)$	$\star$	32
	STNFS	–	–	32
Composite n and special p	SexTNFS	32	$\star$	32

- Main difficulty: relation collection in dimension > 2.

[KB16]: T. Kim, R. Barbulescu, Extended tower number field sieve. Crypto'16

Collecting relations in TNFS

- Relation collection: find relations between the elements of $\mathscr{F}$.

More precisely, what does this mean? What is a relation? Who is $\mathscr{F}$?

For TNFS: $R = \mathbb{Z}[\iota]/h(\iota)$

In our computation:

- $n = 6 = 3 \times 2$
- $\deg h = \eta = 3$
- $h = \iota^3 - \iota + 1$

$$
\begin{tikzcd}
 & R[X] \arrow[dl] \arrow[dr] & \\
K_1 \supset R[X]/(X^4+1) \arrow[dr] & & K_2 \supset R[X]/(aX^2+bX+c) \arrow[dl] \\
 & (R/pR)[X]/(aX^2+bX+c) \cong \mathbb{F}_{p^6} &
\end{tikzcd}
$$

Collecting relations in TNFS

For TNFS: $R = \mathbb{Z}[\iota]/h(\iota)$

In our computation:

- $n = 6 = 3 \times 2$
- $\deg h = \eta = 3$
- $h = \iota^3 - \iota + 1$

- Relation collection: find relations between the elements of $\mathscr{F}$.

More precisely, what does this mean? What is a relation? Who is $\mathscr{F}$?

$$\begin{tikzcd}[column sep=small]
& R[X] \arrow[dl, "X\mapsto\alpha_1"'] \arrow[dr, "X\mapsto\alpha_2"] & \\
K_1 \supset R[X]/(X^4+1) \arrow[dr] & & K_2 \supset R[X]/(aX^2+bX+c) \arrow[dl] \\
& (R/pR)[X]/(aX^2+bX+c) \cong \mathbb{F}_{p^6} &
\end{tikzcd}$$

Collecting relations in TNFS

- Relation collection: find relations between the elements of $\mathscr{F}$.

More precisely, what does this mean? What is a relation? Who is $\mathscr{F}$?

For TNFS: $R = \mathbb{Z}[\iota]/h(\iota)$

In our computation:

- $n = 6 = 3 \times 2$
- $\deg h = \eta = 3$
- $h = \iota^3 - \iota + 1$

$$
\begin{tikzcd}[column sep=small]
 & R[X] \arrow[dl] \arrow[dr] & \\
K_1 \supset R[X]/(X^4+1) \arrow[dr, "(\mathrm{mod}\ p, aX^2+bX+c)"'] & & K_2 \supset R[X]/(aX^2+bX+c) \arrow[dl, "(\mathrm{mod}\ p)"] \\
 & \mathbb{F}_{p^3}[X]/(aX^2+bX+c) \cong \mathbb{F}_{p^6} &
\end{tikzcd}
$$

Collecting relations in TNFS

- Relation collection: find relations between the elements of $\mathscr{F}$.

More precisely, what does this mean? What is a relation? Who is $\mathscr{F}$?

For TNFS: $R = \mathbb{Z}[\iota]/h(\iota)$

In our computation:

- $n = 6 = 3 \times 2$
- $\deg h = \eta = 3$
- $h = \iota^3 - \iota + 1$

$$
\begin{tikzcd}
 & R[X] \arrow[dl] \arrow[dr] \arrow[dd, bend right] \arrow[dd, bend left] & \\
K_1 \supset R[X]/(X^4+1) \arrow[dr] & & K_2 \supset R[X]/(aX^2+bX+c) \arrow[dl] \\
 & \mathbb{F}_{p^6} &
\end{tikzcd}
$$

Collecting relations in TNFS: what is a relation?

$$R = \mathbb{Z}[\iota]/(\iota^3 - \iota + 1)$$

$$\phi(\iota, X) = a(\iota) - b(\iota)X \in R[X]$$

$$\begin{tikzcd}
 & R[X] \arrow[dl] \arrow[dr] & \\
K_1 \supset R[X]/(X^4+1) \arrow[dr] & & K_2 \supset R[X]/(aX^2+bX+c) \arrow[dl] \\
 & \mathbb{F}_{p^6} &
\end{tikzcd}$$

$$\phi(\iota, \alpha_1) = a(\iota) - b(\iota)\alpha_1$$

Test $N(\phi(\iota, \alpha_1))$ for B-smoothness:

$\longrightarrow$ prime factors smaller than B

Equality in finite field = Relation

Collecting relations in TNFS: what is a relation?

- Relation collection: find relations between the elements of $\mathscr{F}$.

$$
\begin{tikzcd}
 & R[X] \arrow[dl] \arrow[dr] & \\
K_1 \supset R[X]/(X^4+1) \arrow[dr] & & K_2 \supset R[X]/(aX^2+bX+c) \arrow[dl] \\
 & \mathbb{F}_{p^6} &
\end{tikzcd}
$$

Who is $\mathscr{F}$?

Prime ideals of small norm in the ring of integers of the intermediate number fields

$$\prod \mathfrak{p}_i^{e_i} \text{ " = " } \prod \mathfrak{q}_j^{f_j}$$

Collecting relations in TNFS: what is a relation?

And to solve a linear system …

$$\begin{tikzcd}
 & R\left[X\right] \arrow[dl] \arrow[dr] & \\
K_1 \supset R\left[X\right]/(X^4+1) \arrow[dr] & & K_2 \supset R\left[X\right]/(aX^2+bX+c) \arrow[dl] \\
 & \mathbb{F}_{p^6} &
\end{tikzcd}$$

$$\sum e_i \log \mathfrak{p}_i \;\text{" = "}\; \sum f_j \log \mathfrak{q}_j$$

Virtual logarithms!

Collecting relations in TNFS

Relation collection looks for a set of linear polynomials $\phi(\iota, X) = a(\iota) - b(\iota)X \in R[X]$

1. with bounded coefficients $\longrightarrow$ $c \in \mathcal{S}$ where $\mathcal{S}$ is known as the sieving region.

2. such that $N_i(a(\iota) - b(\iota)\alpha_i)$ is B-smooth $\longrightarrow$ Norms divisible only by primes smaller than B:
$c \in$ intersection of suitably constructed lattices $\mathcal{L}$

Concretely, let:

$$a(\iota) = a_0 + a_1\iota + a_2\iota^2$$
$$b(\iota) = b_0 + b_1\iota + b_2\iota^2$$

Goal: find vectors $c = (a_0, a_1, a_2, b_0, b_1, b_2) \in \mathbb{Z}^6$ such that

A new sieving region

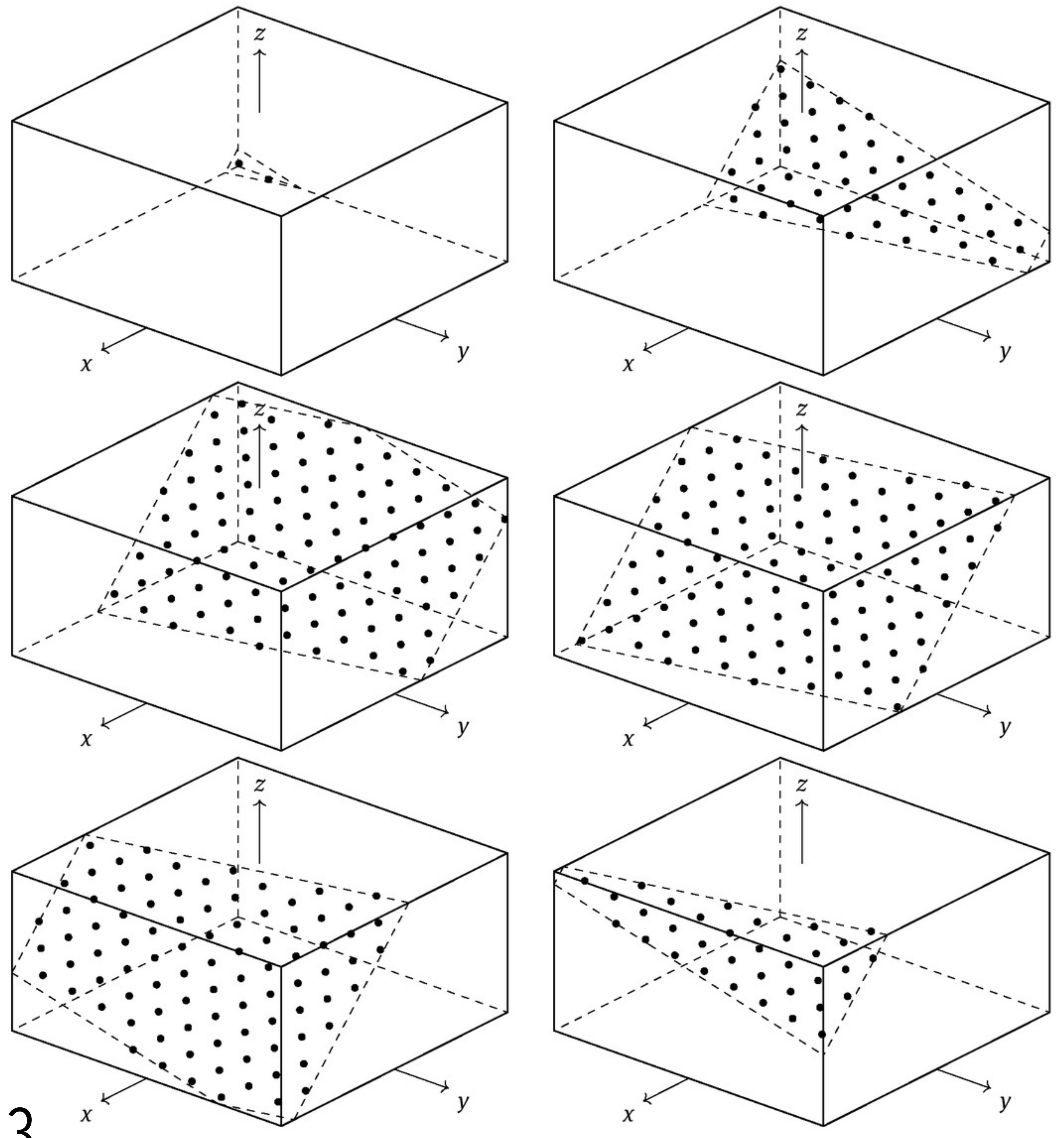
Goal: find $c \in \mathcal{S} \cap \mathcal{L}$

What is the dimension of $\mathcal{S}$? $d = 2\eta = 6$

In previous works:

- For NFS in dimension 2, we look for $(a, b) \in \mathbb{Z}^2$: Franke-Kleinjung's algorithm (2005)
- For NFS in dimension > 2:
  - Grémy's transition-vector algorithm (2017)
  - McGuire and Robinson's hyperplane enumeration (2020)

They all consider: $\mathcal{S} = d$-rectangle



[MC20], sieving in dimension 3

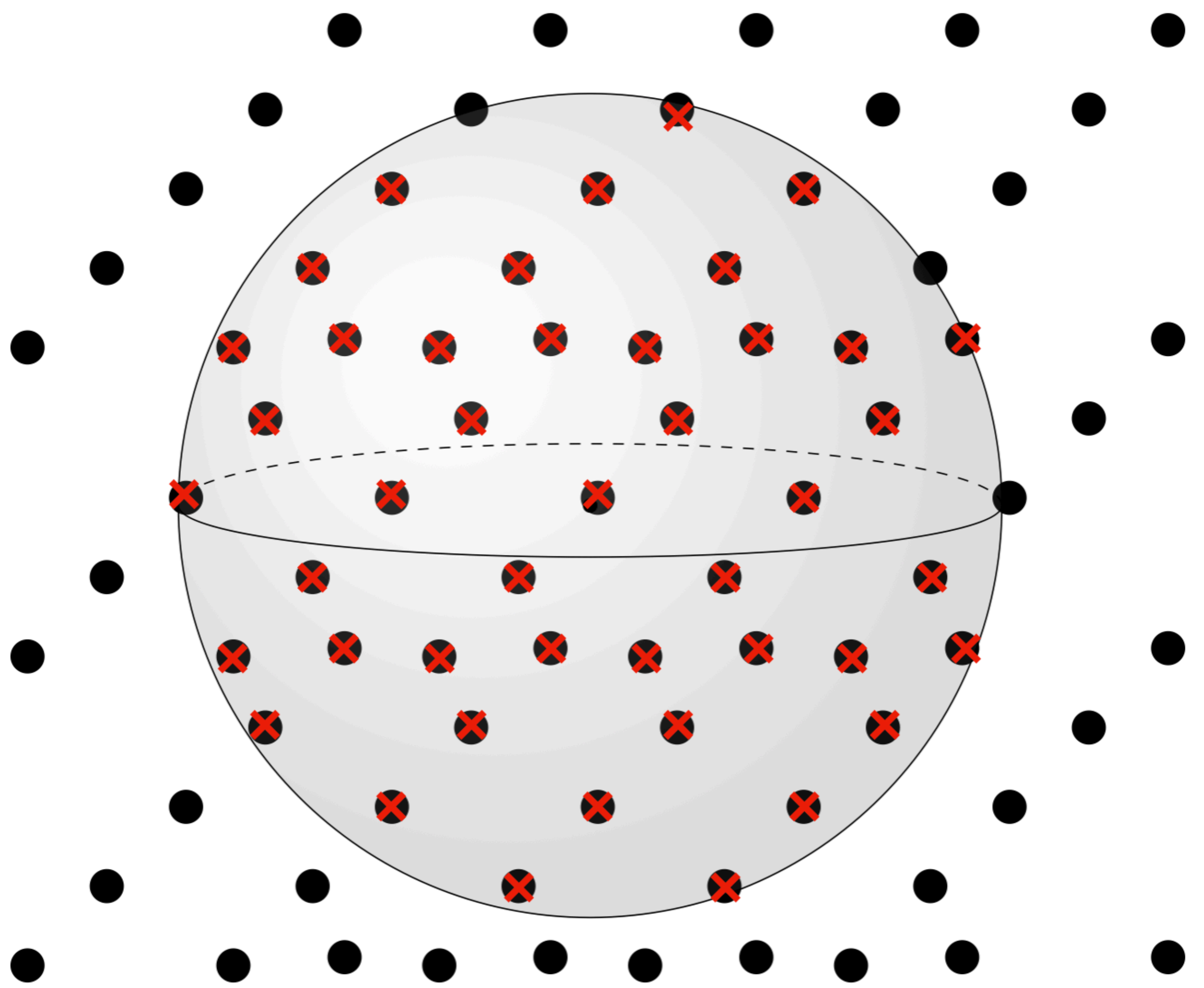
A new sieving region

Goal: find $c \in \mathcal{S} \cap \mathcal{L}$

What is the dimension of $\mathcal{S}$? $d = 2\eta = 6$

In previous works:

- For NFS in dimension 2, we look for $(a, b) \in \mathbb{Z}^2$: Franke-Kleinjung's algorithm (2005)
- For NFS in dimension > 2:
  - Grémy's transition-vector algorithm (2017)
  - McGuire and Robinson's hyperplane enumeration (2020)



They all consider: $\cancel{\mathcal{S} = d}$ ~~-rectangle~~

We look at TNFS so dimension > 2 (since $\eta \geq 2$) and $\mathcal{S} = 6$-**sphere** **(ℓ_2-norm).**

Why do we choose a d-sphere?

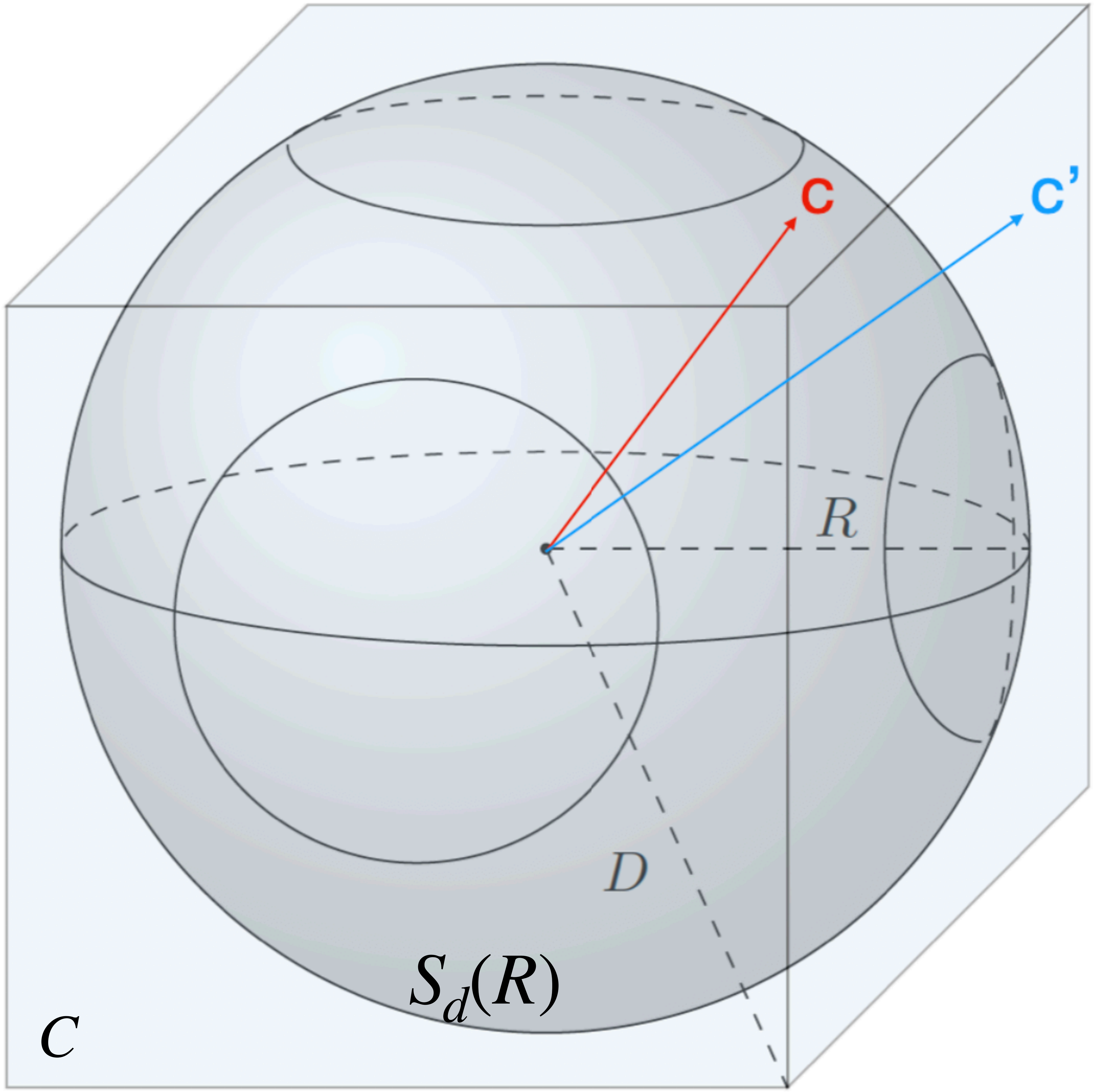
Assumption: *size of norms depends only on size of vector coordinates.*

The norm for $c' \in C \backslash S_d(R)$ is greater than the norm for $c \in S_d(R)$.

When $d \to \infty$:

Difference in norms increases!

Conclusion: choosing $S_d(R)$ leads to smaller norms.



Enumerating in $\mathscr{S} \cap \mathscr{L}$

- Concretely what is $\mathscr{L}$?

  A lattice that describes the divisibility of the ideals by an ideal $\mathfrak{Q}$, known as a special-q ideal and a prime ideal $\mathfrak{p}$ in the intermediate number fields.

  for many $\mathfrak{p}'s$

- The outputs of the enumeration are thus …

  …vectors corresponding to (a, b) pairs whose norms are divisible by $N(\mathfrak{Q})$ and $N(\mathfrak{p})$.

Why? high probability of B-smoothness

Schnorr-Euchner's enumeration [SE94]

- Input: a lattice basis $\mathbf{b}_1, \cdots, \mathbf{b}_d$
- Output: shortest non-zero lattice vector

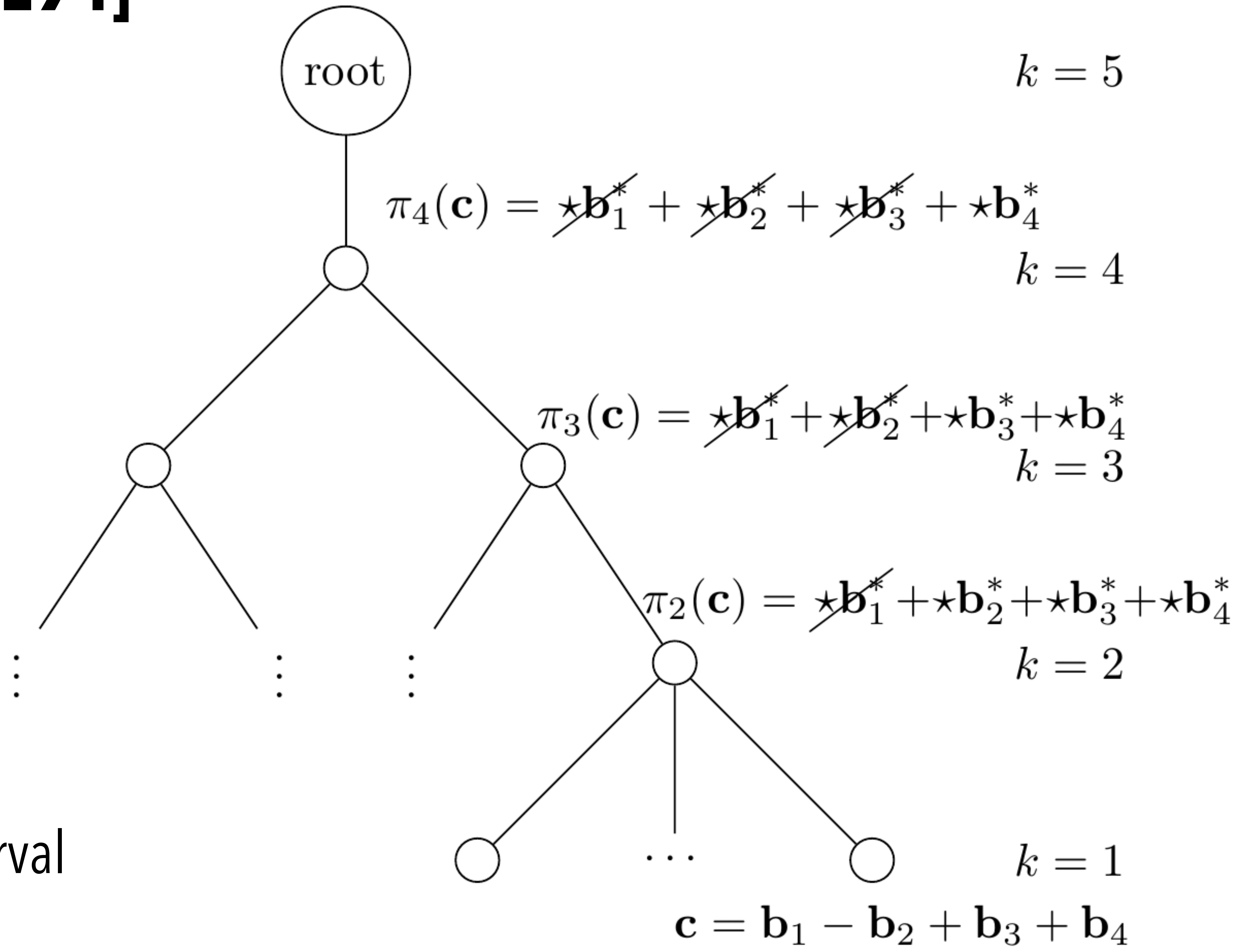
Idea:

1. Construct an enumeration tree

2. Consider projections of the lattice

3. At each level of the tree, enumerate in an interval

4. Depth-first search in the tree



[SE94]: C-P. Schnorr, M. Euchner, Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. Math. Program.'94

Schnorr-Euchner's enumeration [SE94]

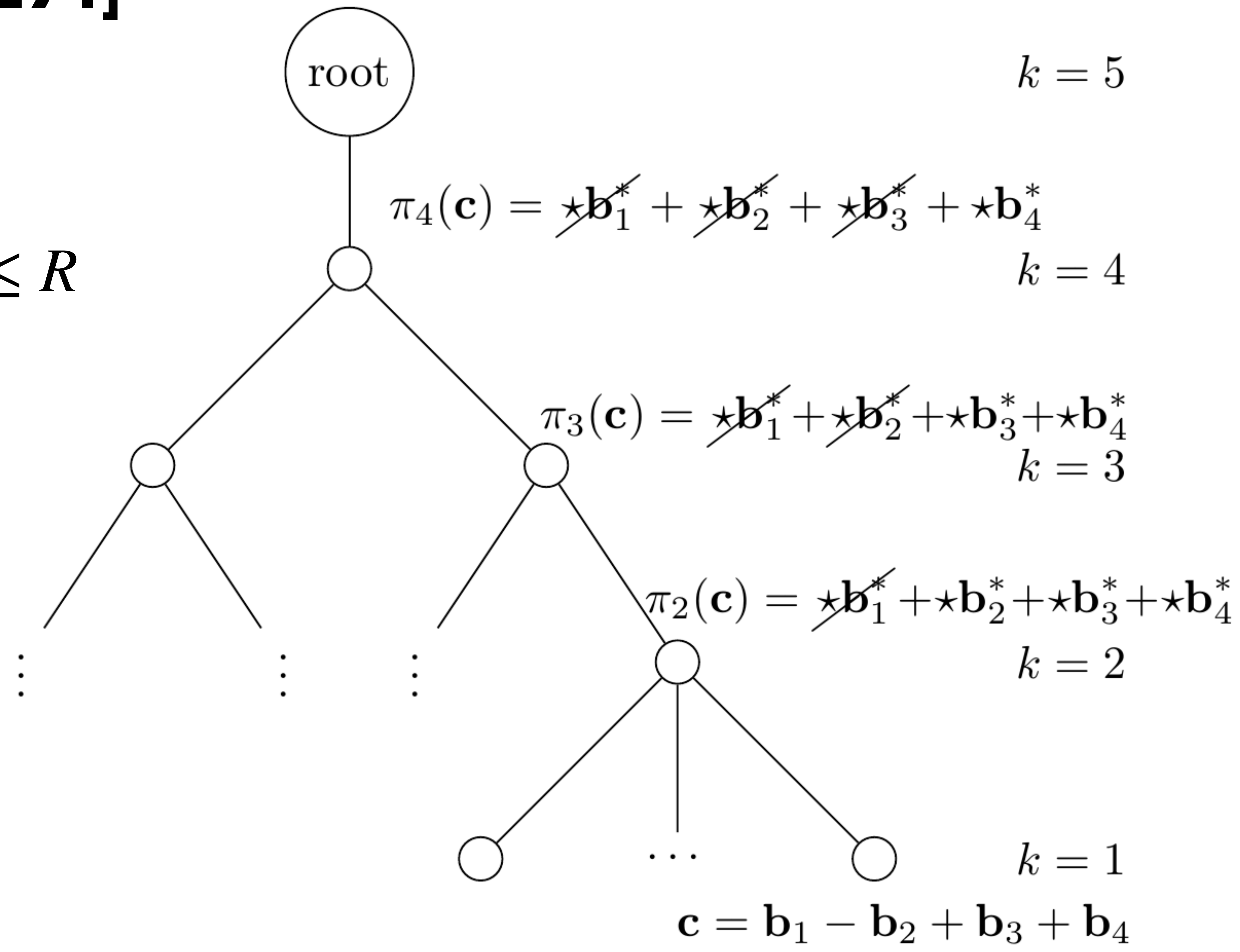
- Input: a lattice basis $\mathbf{b}_1, \cdots, \mathbf{b}_6$
- Output: vectors $c = \sum v_i \mathbf{b}_i$ such that $||c|| \leq R$

Idea:

1. Construct an enumeration tree

2. Consider projections of the lattice

3. Exhaustive search of the coefficients v_i



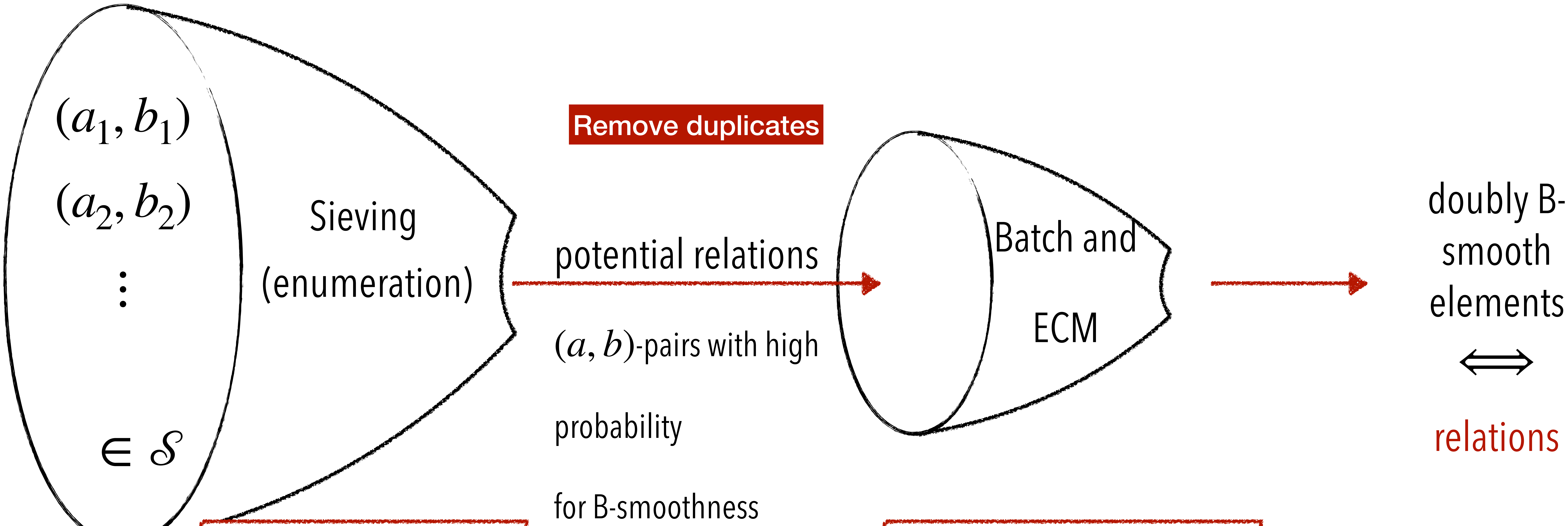
Adapting enumeration to TNFS

- We don't want the shortest non-zero vector but all the vectors of norm smaller than a radius R.
- We optimize the computation of the vector c by reducing the number of computations:

$$c = \sum_{i=1}^{t-1} v_i \mathbf{b}_i + \text{common\_part} \qquad \text{with common\_part} = \sum_{i=t}^{6} v_i \mathbf{b}_i$$

- This gives a $10\ \%$ improvement in our total sieving time.

Relation collection all together



idea: find (a, b)-pairs in $\mathcal{S}$ intersect many lattices $\mathcal{L}_{\mathfrak{p}}$.

idea: test B-smoothness of the potential relations

Removing duplicates

What is a duplicate relation?

Definition: A duplicate relation refers to a pair (a, b) such that there exists another pair (a', b') that leads to the same relation.

Three types of duplicates:

1. Special-q-duplicates $\approx 30\,\%$

$$\left.\begin{array}{l}\text{2. } K_h\text{-unit-duplicates} \\ \text{3. } \zeta_2\text{-duplicates}\end{array}\right\} \approx 54\,\%$$

Are they common? Yes !

Removing duplicates

What is a duplicate relation?

Definition: A duplicate relation refers to a pair (a, b) such that there exists another pair (a', b') that leads to the same relation.

Three types of duplicates:

1. Special-q-duplicates
2. K_h-unit-duplicates
3. ζ_2-duplicates

In NFS:

units: $\{-1,1\}, a > 0.$

$\mathrm{gcd}(a, b) =_? 1$

Identifying and removing duplicates

We provide a new method to identify and remove duplicates in the context of TNFS.

1. K_h-unit-duplicates $a' = ua, b' = ub$ for $u \in O_{K_h}^{\times}$
2. ζ_2-duplicates $a' = \lambda a, b' = \lambda b$ for $\lambda \in O_{K_h}$

Identification: compute $k = \dfrac{a}{b} \pmod{h} \in K_h$ and store k in a hash table.

Warning 1: we want to keep the smallest pair!

Keeping a primitive pair

Why? keeping $(\lambda a, \lambda b) \Rightarrow$ extra ideals in the prime ideal decomposition $\Rightarrow$ extra coefficients in the matrix

In our work: new algorithm for ζ_2-duplicates based on a gcd computation of norms.

Input: (a, b)-pair

Output: primitive (a, b)-pair

Idea: check if $\gcd(N_1(a,b), N_2(a,b)) =_? 1$

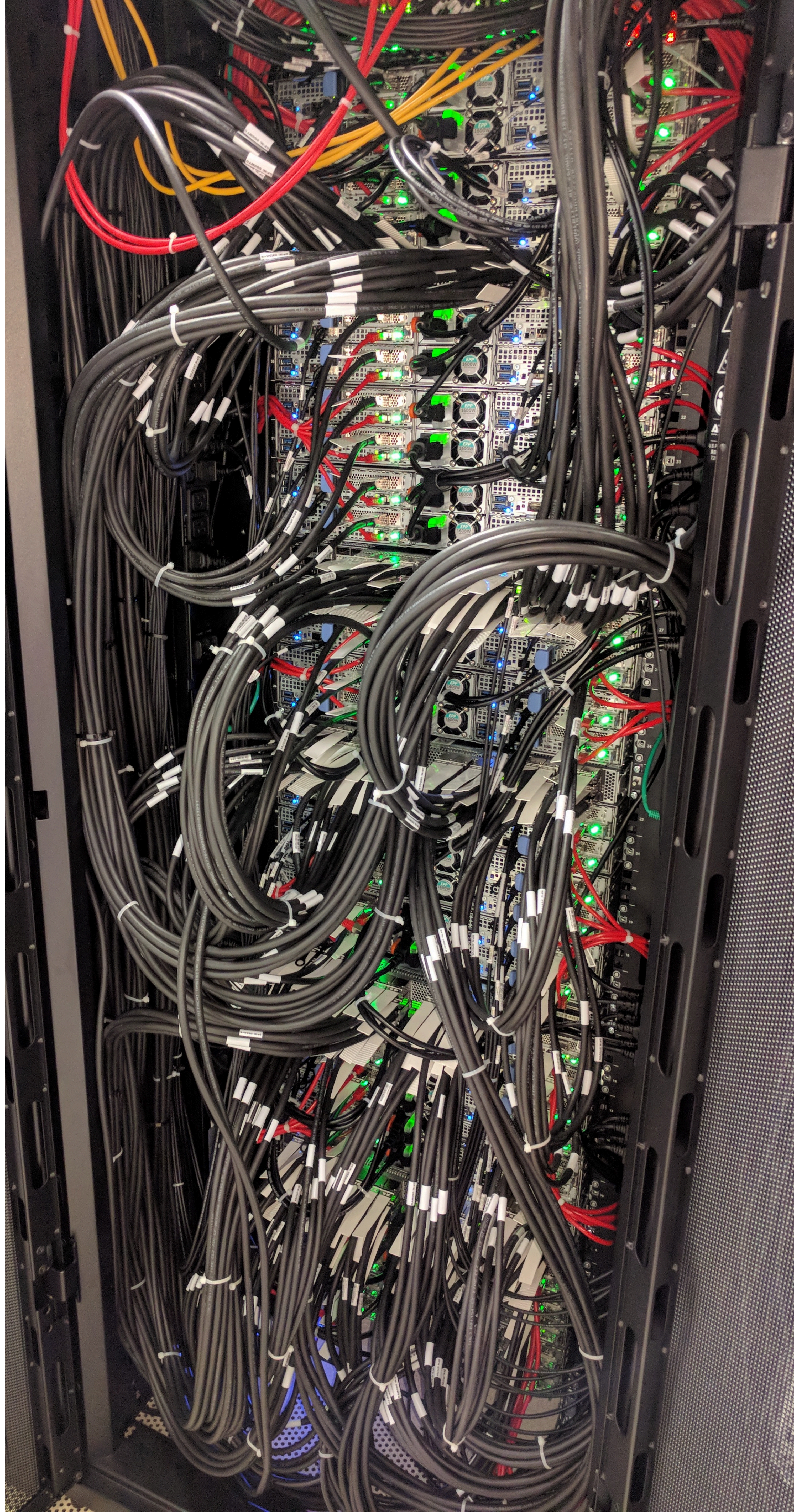
Warning 2: it doesn't work for K_h-unit-duplicates!

What we needed for a record computation

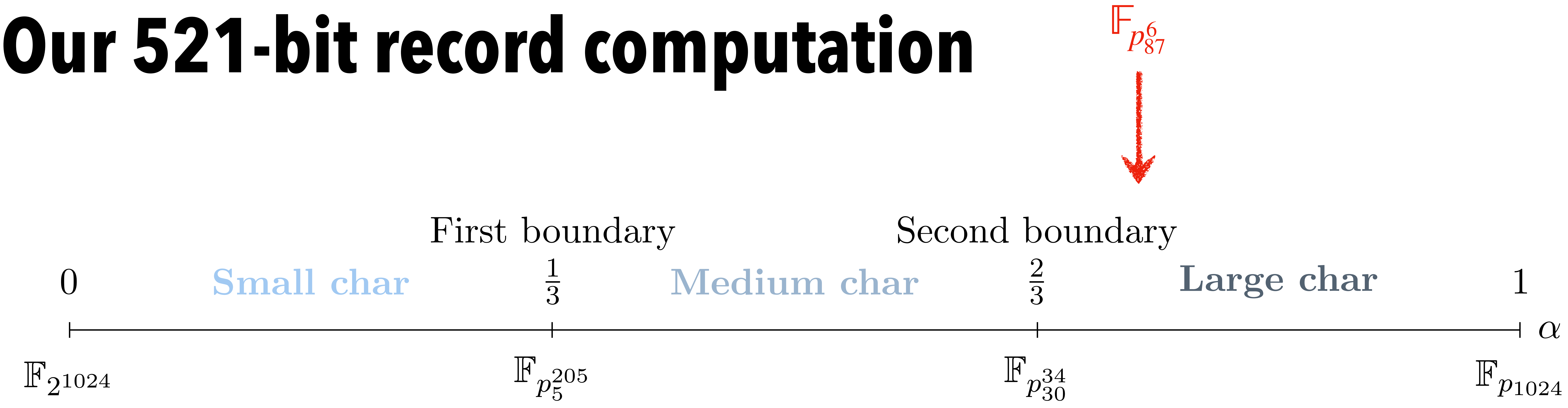
- A fast sieving algorithm in dimension > 2.
- Identifying and removing duplicate relations.
- Adapting Schirokauer maps (virtual logarithms) to TNFS context.
- Glue-code to branch into CADO-NFS.
- A nice target: $\mathbb{F}_{p^6}$.

↑
in theory…

in practice… ⟶ grvingt



Our 521-bit record computation



Total computation time (core hours):

Relation Collection	Linear algebra	Schirokauer maps	Descent	Overall time
23,300	1,403	40	55	24,798

Focus on relation collection:

Parameters	[GGMT17]	[MR21]	This work
Algorithm	NFS	NFS	TNFS
Field size (bits)	422	423	521
Sieving dimension	3	3	6
Sieving time	201,600	69,120	**23,300**

[GGMT17]: L. Grémy, A. Guillevic, F. Morain, E. Thomé, Computing discrete logarithm in Fp6. Sac'17
[MR21]: G. McGuire, O. Robinson, Lattice Sieving in three dimensions for discrete log in medium characteristic. Journal of mathematical cryptology'21

A discrete logarithm

Finite field: $\mathbb{F}_{p^6}$ with 87-bit prime p, generator $g = x + \imath$

$$\begin{aligned}\text{target} = {} & (31415926535897932384626433 + 83279502884197169399375105\imath \\ & +82097494459230781640628620\imath^2) + x(89986280348253421170679821 \\ & +48086513282306647093844609\imath + 55058223172535940812848111\imath^2)\end{aligned}$$

$$\log(\text{target}) = 76272808168753222977667479701383785303538529763154 98$$

Summary of contributions

How can we assess the security of protocols in which a modular exponentiation involving a secret exponent is performed?

- Looking at implementation vulnerabilities during fast exponentiation.
  - → Summary of key recovery methods from partial information
  - → Two concrete attacks on signing algorithms using lattice techniques and partial information
- Estimate the hardness of DLP in the groups considered by the protocols.
  - → Asymptotic complexity analysis for pairing-related finite fields
  - → First implementation of the variant TNFS and record-computation

Perspectives

Concerning DLP:

- Using Galois automorphisms to improve sieving and linear algebra.
- A new target $\mathbb{F}_{p^{12}}$ and sieving in dimension 8.
- Towards the implementation of other variants: considering Multiple NFS?

Concerning key recovery methods:

- Recoving RSA private key d from MSB of d.

Thank you for your attention!

Additional slides

Related publications

1. Hardness of DLP for $\mathbb{F}_{p^n}$

- *Asymptotic complexities of discrete logarithm algorithms in pairing-relevant finite fields,* with Pierrick Gaudry and Cécile Pierrot, at **Crypto 2020**
- *Lattice enumeration for Tower NFS: a 521-bit discrete logarithm computation,* with Pierrick Gaudry and Cécile Pierrot, **submitted**

2. Implementation vulnerabilities

- *Recovering cryptographic keys from partial information, by example,* with Nadia Heninger, **Eprint 2020/1507**
- *CacheQuote: Efficiently Recovering Long-term Secrets of SGX EPID via Cache attacks,* with Fergus Dall, Thomas Eisenbarth, Daniel Genkin, Nadia Heninger, Ahmad Moghimi and Yuval Yarom, at **CHES 2018**
- *A Tale of Three Signatures: practical attack of ECDSA with wNAF,* with Cécile Pierrot and Rémi Piau, at **Africacrypt 2020**

A discrete logarithm (in more details)

$p =$ 0x6fb96ccdf61c1ea3582e57 (87-bit prime) $\quad n = 6$

$\mathbb{F}_{p^6} = \mathbb{F}_{p^3}[x]/(x^2 + 644177233069914644196222353x + 1)$

Irreducible factor mod p, here f2

target = $a(\imath) + xb(\imath) \in \mathbb{F}_{p^6}$ with: $a(\imath), b(\imath)$ of degree 2 and coefficients $< p$.

$$\text{target} = (31415926535897932384626433 + 83279502884197169399375105\imath$$
$$+82097494459230781640628620\imath^2) + x(89986280348253421170679821$$
$$+48086513282306647093844609\imath + 55058223172535940812848111\imath^2)$$

generator = $x + \imath$

$\log(\text{target}) = 7627280816875322297766747970138378530353852976315498$

Verification: $(x + \imath)^{\log(\text{target})} = \text{target} \pmod{\ell\text{-th powers}}$

Choice of subgroup

Initial target: $\mathbb{F}_{p^6}$ $\xrightarrow{\text{Pohlig-Hellman:}}$ Prime order subgroup of order $\ell \mid p^6 - 1$

We have the following factorisation: $p^6 - 1 = (p-1)(p+1)(p^2+p+1)(p^2-p+1)$

- $p - 1 = |\mathbb{F}_p^\times|$ If g and h are of order $\ell \mid p-1 \Rightarrow g, h \in \mathbb{F}_p^\times \Rightarrow$ NFS in $\mathbb{F}_p$ of 87 bits
- $p + 1 = |\mathbb{F}_{p^2}^\times| / |\mathbb{F}_p^\times|$ If g and h are of order $\ell \mid p+1 \Rightarrow g, h \in \mathbb{F}_{p^2}^\times \Rightarrow$ NFS in $\mathbb{F}_{p^2}$ of 175 bits
- $p^2 + p + 1 = |\mathbb{F}_{p^3}^\times| / |\mathbb{F}_p^\times|$ If g and h are of order $\ell \mid p^2+p+1 \Rightarrow g, h \in \mathbb{F}_{p^3}^\times \Rightarrow$ NFS in $\mathbb{F}_{p^3}$ of 261 bits
- $p^2 - p + 1$: 6th-cyclotomic subgroup Here, we can't go in a smaller subgroup…

Attention: it is not the largest subgroup!

Multiplicative group of a finite field

- The non-zero elements of a finite field form a multiplicative group.

- This group is cyclic, so all non-zero elements can be expressed as powers of a single element called a primitive element of the field.

Example 1: prime order finite fields: $\mathbb{F}_p \cong \mathbb{Z}/p\mathbb{Z}$

multiplicative group: $\mathbb{F}_p^\times = \{1,2,\cdots,p-1\} = \mathbb{F}_p \backslash \{0\}$

Example 2: non-prime order finite fields: $\mathbb{F}_{p^n} \cong \mathbb{F}_p[X]/(P)$

--> elements are polynomials over $\mathbb{F}_p$ whose degree is less than n.

multiplicative group: $\mathbb{F}_{p^n}^\times = \{\text{invertible polynomials}\} = \mathbb{F}_{p^n} \backslash \{0\}$

Number field vs Function fields

Number field:

Finite extension of $\mathbb{Q}$

$\mathbb{Q} = \{p/q : p, q \text{ integers}\}$

$K = \mathbb{Q}[x]/(f)$

Example: $f = x^2 - d$

$K = \{x + y\sqrt{d} : x, y \in \mathbb{Q}\}$

Factor basis: prime ideals in $\mathcal{O}_K$

B-smoothness: compute norm of ideal = integer (from a resultant)

Function field:

Finite extension of $\mathbb{F}_p(\imath)$

$\mathbb{F}_p(\imath) = \{p(\imath)/q(\imath) : p(\imath), q(\imath) \in \mathbb{F}_p[\imath]\}$

$K = \mathbb{F}_p(\imath)[x]/(f)$

Example: $f = x^2 - (\imath^3 + 2\imath - 3)$

$K = \{x_0 + x_1\sqrt{\imath^3 + 2\imath - 3} : x_0, x_1 \in \mathbb{F}_p(\imath)\}$

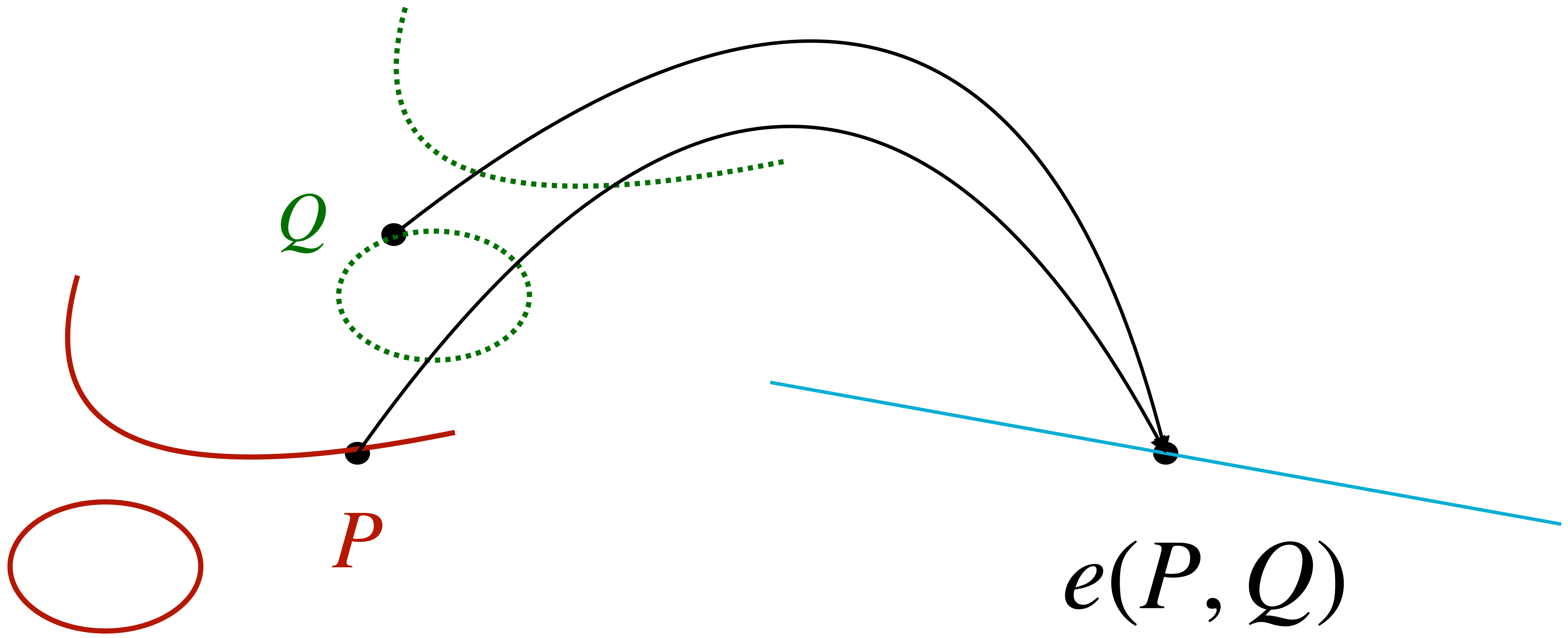
Factor basis: prime ideals in $\mathcal{O}_K$

B-smoothness: compute norm of ideal = univariate polynomial (from a bivariate resultant)

Cryptographic pairings

A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$

For cryptography $e : \mathscr{E}(\mathbb{F}_p) \times \mathscr{E}(\mathbb{F}_{p^k}) \rightarrow \mathbb{F}_{p^k}$



k: embedding degree

Pairing-friendly curves

Elliptic curves for which the pairing is efficiently computable

–> it contains a subgroup of order r whose embedding degree k is not too large, which means that computations in the field $\mathbb{F}_{p^k}$ are feasible.

Example: BN curves, BLS curves

Security of pairing-friendly curves

The security of pairing-friendly curves is evaluated by the hardness of DLP over $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$.

Over elliptic curves: square-root complexity $\mathcal{O}(\sqrt{r})$

Over finite fields: cost of computing discrete logs in $\mathbb{F}_{p^k}$: complexity of exTNFS !

For 128-bit level of security:

Menezes, Sarkar, Singh [MSS17]: minimum bit length of p of BN curves is 383 bits and for BLS12 curves is 384 bits.

For 256-bit level of security:

Kiyomura et al. [KIK17]: minimum bit length of p^k of BLS48 curves as 27,410 bits, i.e., 572 bits of p.

Guillevic’s blogpost

https://members.loria.fr/AGuillevic/pairing-friendly-curves/#pairing-friendly-curves-at-the-128-bit-security-level

> For efficient non-conservative pairings, choose BLS12-381 (or any other BLS12 curve or Fotiadis–Martindale curve of roughly 384 bits), for conservative but still efficient, choose a BLS12 or a Fotiadis–Martindale curve of 440 to 448 bits.

Efficient pairing:

Barreto–Lynn–Scott BLS12-381:

- $k = 12$
- p: 381 bits
- p^k: 4569 bits

"Conservative" pairing:

Barreto–Lynn–Scott BLS12-440:

- $k = 12$
- p: 440 bits
- p^k: 5280 bits

Examples: implementations of pairing-friendly curves

- Zcash: BLS12-381
- Ethereum 2.0: BLS12-381, BN curves with 254 bits of p (CurveFp254BNb) and 382 bits of p (CurveFp382_1 and CurveFp382_2)

Schnorr-Euchner's enumeration

We want $c = \sum_{i=1}^{d} v_i \mathbf{b}_i$ such that $||c|| \leq R$

Level $k = i$

$\cdots$

Leaf : $k = 1$

Internal node corresponds to:

$$\pi_k(c) = \sum_{j=k}^{d} \left(v_j + \sum_{i=j+1}^{d} (\mu_{i,j} v_i) \mathbf{b}_j^* \right)$$

Example: for k =2

$$\pi_2(c) = -\mathbf{b}_2^* + 3\mathbf{b}_3^* - 5\mathbf{b}_4^*$$

IF $||\pi_k(c)|| \geq R$

Go up one level and vary v_{i+1}

Example: now k =3

$$\pi_3(c) = 3\mathbf{b}_3^* - 5\mathbf{b}_4^*$$

$$\pi_3(c') = v_3\mathbf{b}_3^* - 5\mathbf{b}_4^*$$

Ignore the subtree

More precisely …

Internal node corresponds to:

Level $k = i$

$$\pi_k(c) = \sum_{j=k}^{d} \left(v_j + \sum_{i=j+1}^{d} (\mu_{i,j} v_i) \mathbf{b}_j^* \right)$$

IF $||\pi_k(c)|| \leq R$

$\downarrow$

Explore the subtree and vary v_{i-1}

…

Exhaustive search of all coefficients v_i for $i = 1, \cdots, d$

Leaf : $k = 1$

Leaves: all vectors $c = \sum_{i=1}^{d} v_i \mathbf{b}_i$ such that $||c|| \leq R$

Relation collection all together

Algorithm 8 Relation collection for a given special-q with sieving, batch and ECM

Input: A prime ideal $\mathfrak{Q}$, a sieving region $\mathcal{S}$

Output: A list of relations.

1: Construct the lattice $\mathcal{L}_{\mathfrak{Q}}$ and LLL-reduce it.
2: **for** each prime ideal $\mathfrak{p}$ in K_1 (or K_2) up to $p_{\max}$ **do**
3: Construct the lattice $\mathcal{L}_{\mathfrak{Q},\mathfrak{p}}$
4: Enumerate all vectors in $\mathcal{L}_{\mathfrak{Q},\mathfrak{p}} \cap \mathcal{S}$.
5: For each vector enumerated, keep track of the size of the factors p with a sieving table.
6: For promising vectors, compute approximations of the norms N_1, N_2 and identify sieve-survivors.
7: Remove duplicates.
8: Run batch algorithm with input the norms N_1 and N_2 of the sieve-survivors and primes up to p_{batch}. Keep batch-survivors.
9: Run ECM on the batch-survivors.
10: **return** Vectors with doubly-B-smooth norms which give relations as selected by ECM.

Keeping a primitive pair: algorithm

We provide an algorithm for ζ_2-duplicates based on a gcd computation that takes an (a, b)-pair and transforms it into its primitive version.

Idea: check if $\gcd(N_1(a,b), N_2(a,b)) =_? 1$

Yes: the pair is primitive, we keep it.

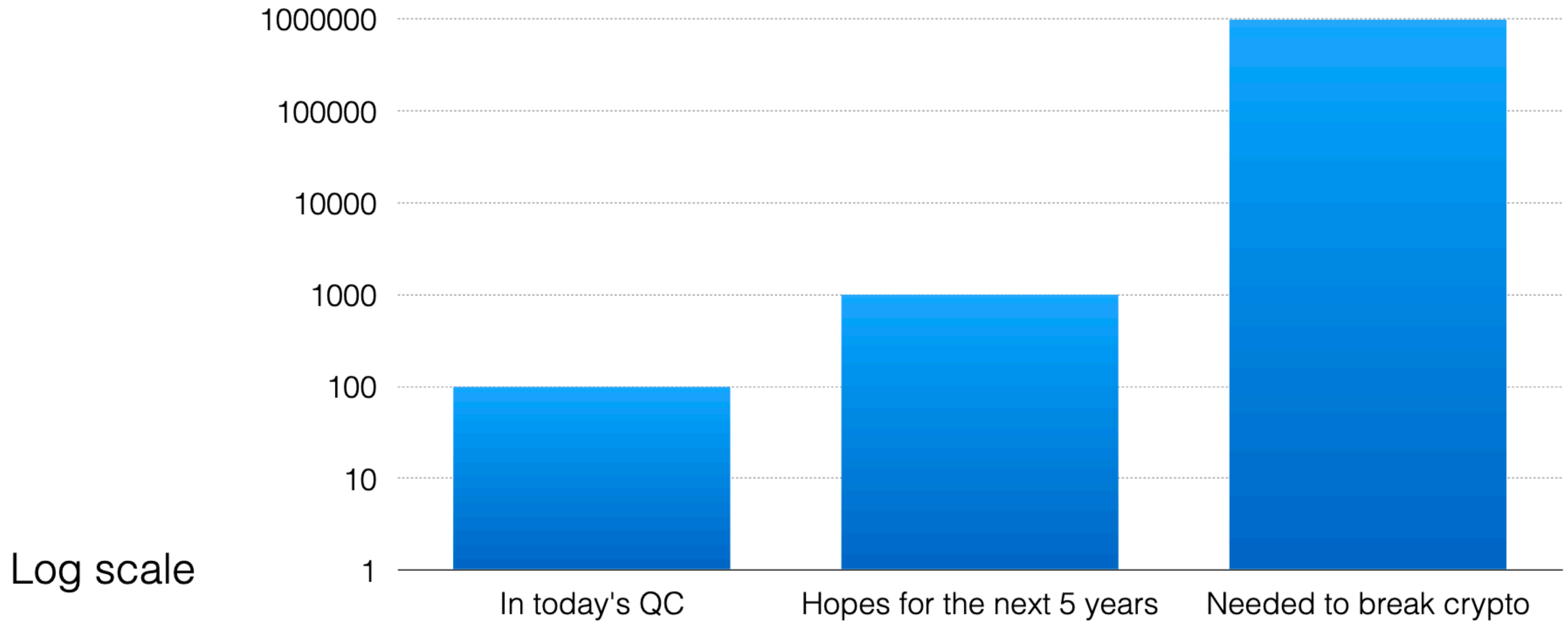
No: for each prime $\ell \mid \gcd(N_1(a,b), N_2(a,b))$

Find $\beta \in O_{K_h}$ such that $a/\beta, b/\beta \in O_{K_h}$ Warning 2: doesn't necessarily exist

Recompute $\gcd(N_1(a/\beta, b/\beta), N_2(a/\beta, b/\beta))$

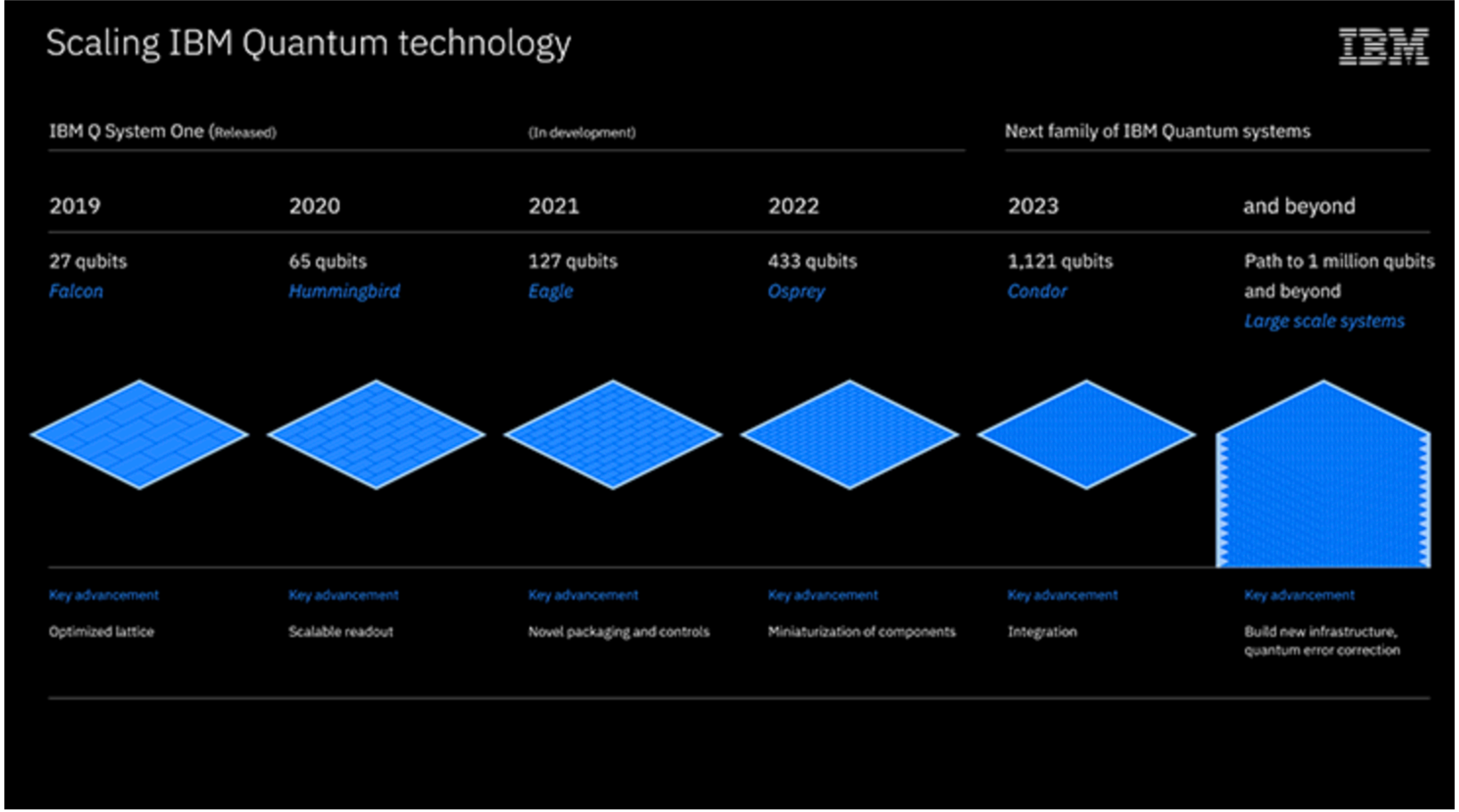
Warning 3: it doesn't work for K_h-unit-duplicates!

How many qubits in a quantum computer?



From JP Aumasson

How many qubits in a quantum computer?



"In 2023, we will debut the 1,121-qubit IBM Quantum Condor processor…"

"as we explore realms even further beyond the thousand qubit mark, today's commercial dilution refrigerators will no longer be capable of effectively cooling and isolating such potentially large, complex devices."

"super-fridge"

[Jay Gambetta]
IBM Fellow and Vice President, IBM Quantum