Higher dimensional sieving

for the number field sieve algorithms

Laurent Grémy

July 20th, 2018 – -XIII, Madison (Wisconsin)

Univ Lyon, CNRS, ENS de Lyon, Inria, Université Claude Bernard Lyon 1, LIP

Discrete logarithm problem

Definition

Finite cyclic group $(G, \circ)$, a in G, g a generator of G.

Find k in $\mathbb{N}$,

$$\underbrace{g \circ g \circ \cdots \circ g}_{k \text{ times}} = a.$$

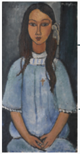
Hard to compute in $\left(\mathbb{F}_{p^n}^*, \ \cdot\right)$

Cryptosystems rely on:

- Diffie–Hellman KE
- DSA signature
- ElGamal encryption
- Triffie–Hellman KE

$(G, \circ)$, g

Alice, k_0 | Bob, k_1



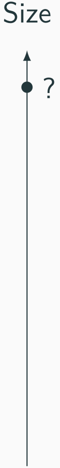
$a = g^{k_0}$ →

← $b = g^{k_1}$

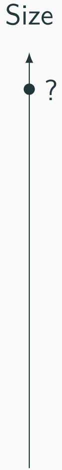
$K = b^{k_0}$ | $K = a^{k_1}$

Algorithm:

1. polynomial selection
   - choose $\varphi = \gcd(f_0, f_1)$ to define $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/\varphi(x)$
2. relation collection
   - find many relations $\sum_i e_i \log b_i = \sum_j e'_j \log b'_j$
3. linear algebra
   - solve linear system
4. individual logarithm
   - compute logarithm of the target



Number field sieve



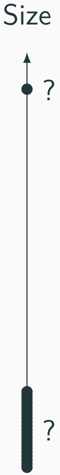
Algorithm:

1. polynomial selection
   - choose $\varphi = \gcd(f_0, f_1)$ to define $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/\varphi(x)$
2. relation collection
   - find many relations $\sum_i e_i \log b_i = \sum_j e'_j \log b'_j$
3. linear algebra
   - solve linear system
4. individual logarithm
   - compute logarithm of the target

Number field sieve

Algorithm:

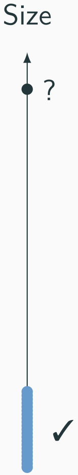
1. polynomial selection
   - choose $\varphi = \gcd(f_0, f_1)$ to define $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/\varphi(x)$
2. relation collection
   - find many relations $\sum_i e_i \log b_i = \sum_j e'_j \log b'_j$
3. linear algebra
   - solve linear system
4. individual logarithm
   - compute logarithm of the target



Number field sieve

Algorithm:

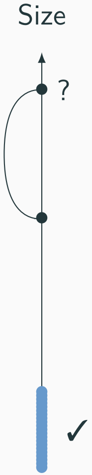
1. polynomial selection
   - choose $\varphi = \gcd(f_0, f_1)$ to define $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/\varphi(x)$
2. relation collection
   - find many relations $\sum_i e_i \log b_i = \sum_j e'_j \log b'_j$
3. linear algebra
   - solve linear system
4. individual logarithm
   - compute logarithm of the target



Number field sieve

Algorithm:

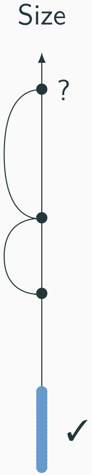
1. polynomial selection
   - choose $\varphi = \gcd(f_0, f_1)$ to define $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/\varphi(x)$
2. relation collection
   - find many relations $\sum_i e_i \log b_i = \sum_j e'_j \log b'_j$
3. linear algebra
   - solve linear system
4. individual logarithm
   - compute logarithm of the target



Number field sieve

Algorithm:

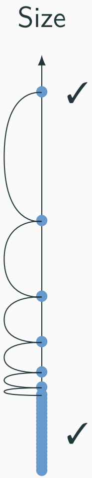
1. polynomial selection
   - choose $\varphi = \gcd(f_0, f_1)$ to define $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/\varphi(x)$
2. relation collection
   - find many relations $\sum_i e_i \log b_i = \sum_j e'_j \log b'_j$
3. linear algebra
   - solve linear system
4. individual logarithm
   - compute logarithm of the target



Number field sieve

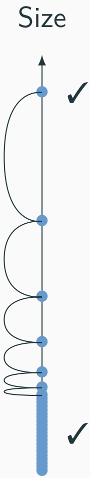
Algorithm:

1. polynomial selection
   - choose $\varphi = \gcd(f_0, f_1)$ to define $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/\varphi(x)$
2. relation collection
   - find many relations $\sum_i e_i \log b_i = \sum_j e'_j \log b'_j$
3. linear algebra
   - solve linear system
4. individual logarithm
   - compute logarithm of the target



Algorithm:

1. polynomial selection
   - choose $\varphi = \gcd(f_0, f_1)$ to define $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/\varphi(x)$
2. **relation collection**
   - find many relations $\sum_i e_i \log b_i = \sum_j e'_j \log b'_j$
3. linear algebra
   - solve linear system
4. individual logarithm
   - compute logarithm of the target



Complexities

L function

$$L_{p^n}(c) = \exp\left[\log(p^n)^{1/3}\log\log(p^n)^{2/3}\right]^{(c^{1/3}+o(1))}$$

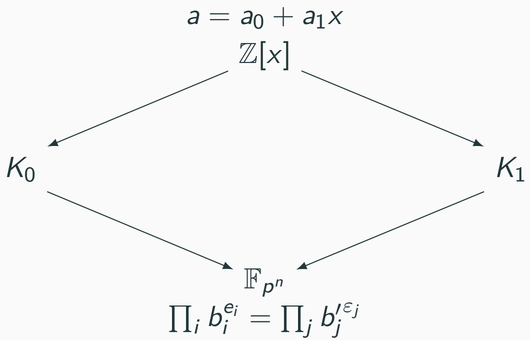
Algorithm	Type of n	Complexity	Reference
NFS	small	$c = 96/9$	[BaGaGrMo'15a]
	tiny	$c = 64/9$	[Schirokauer'92, JoLeSmVe'06,...]
exTNFS	composite	$c = 64/9$	[KiBa'16]

Finite field: gotta catch'em all[1]

Finite field $\mathbb{F}_{p^n}$	Bit size	Cost: CPU days	Reference
$n = 12$	203	11	[HaAoKoTa'13]
$n = 11$	—	—	—
⋮	⋮	⋮	⋮
$n = 7$	—	—	—
$n = 6$	422	9,520	[GrGuMoTh'18]
$n = 5$	324	359	[GrGuMo'17]
$n = 4$	392	510	[BaGaGuMo'15b]
$n = 3$	593	8,400	[GaGuMo'16]
$n = 2$	595	175	[BaGaGuMo'15a]
$n = 1$	768	1,935,830	[KlDiLePrSt'17]

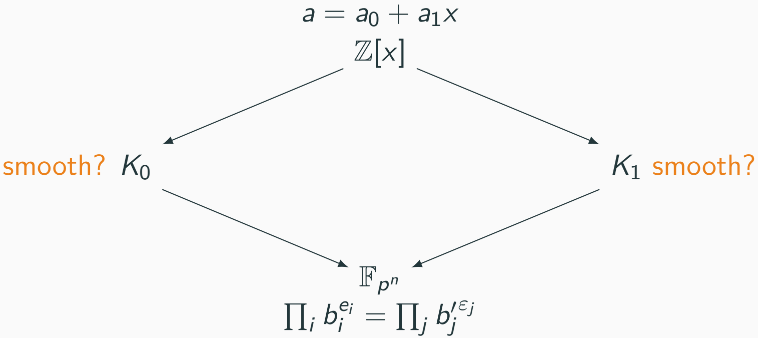
[1] Data extracted from [DLDB]

$K_i = \mathbb{Q}[x]/f_i(x)$

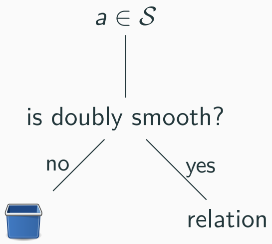


Relation collection for tiny n

$K_i = \mathbb{Q}[x]/f_i(x)$

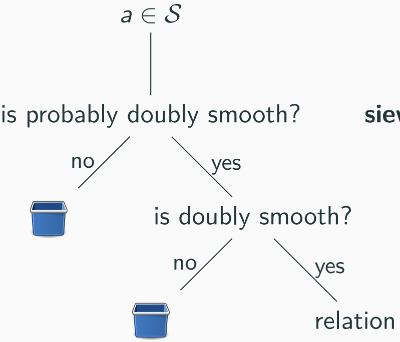


Relation collection: algorithms



Theoretically: factorize all the possible a's

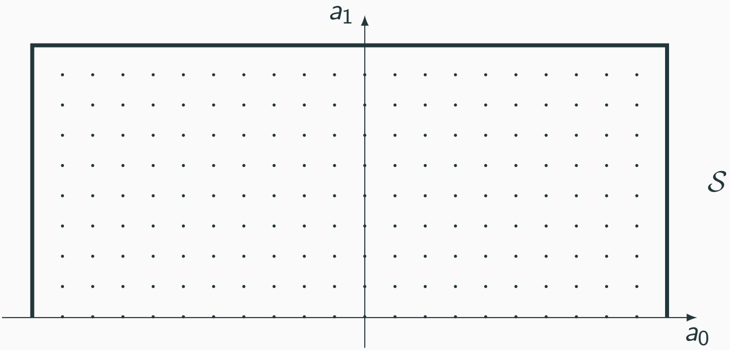
Relation collection: algorithms



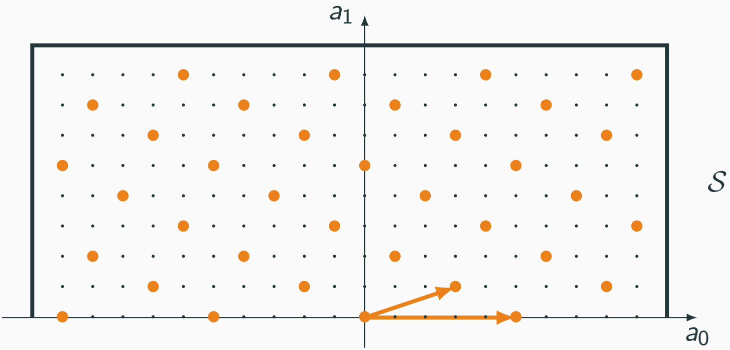
sieve

Practically: factorize only promising a's

Sieving ≡ enumerating lattice points



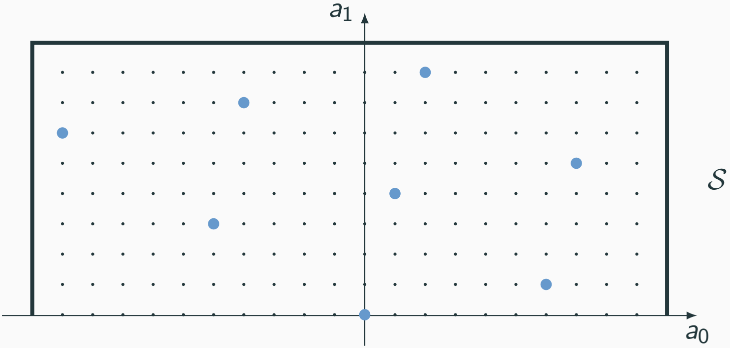
Sieving $\equiv$ enumerating lattice points



Divisible by

$\mathfrak{R}_0$

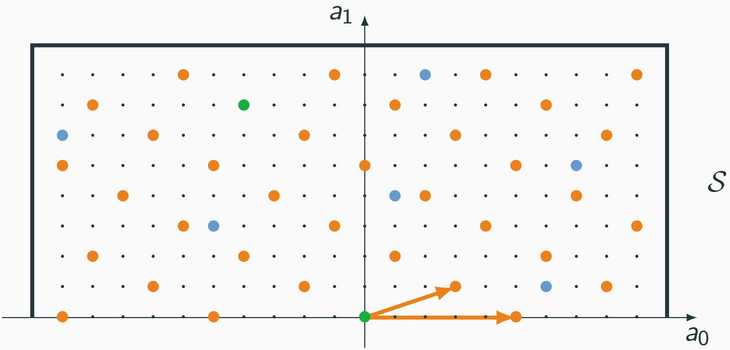
Sieving $\equiv$ enumerating lattice points



Divisible by

$\mathfrak{R}_1$

Sieving $\equiv$ enumerating lattice points



Divisible by

$\mathfrak{R}_0$, $\mathfrak{R}_1$, $\mathfrak{R}_0$ and $\mathfrak{R}_1$

Relation collection: dimension

Algorithm	Type of n	# coeff. of a
NFS	small	$t \geq 2$
	tiny	2
exTNFS	$\eta\kappa$	$2\eta \geq 4$

Sieve algorithms

	Sieve algorithm	Reference
2-dim	line sieve	—
	lattice sieve	[FrKl'05]
3-dim	line sieve	[Zajac'08]
	3-dim lattice	[HaAoKoTa'15]
	plane sieve	[GaGrVi'16]
	space sieve	[GaGrVi'16]
4-dim	line sieve	—
	plane sieve	[CADO-NFS]
	`sparsentv`	**This work**
	`localntv`	**This work**
	`globalntv`	**This work**
⋮	⋮	⋮

Outline of the talk

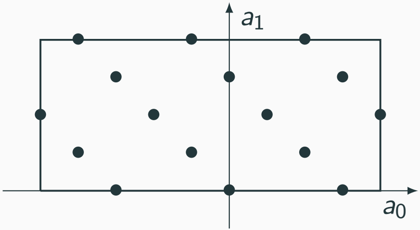
NFS in a nutshell

Generic sieve algorithms

- With oracle
- Without oracle
- Practical results

Conclusion and perspectives

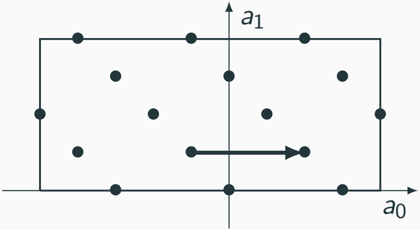
Examples in 2-dim



Using a k-transition vector

- modifies the coordinates 0 to k
- reaches an element with the smallest possible coordinate k

Examples in 2-dim

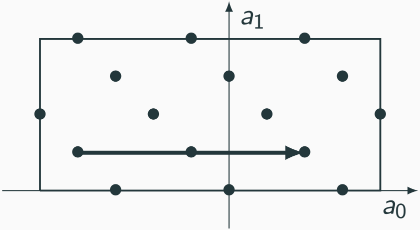


0-transition vector

Using a k-transition vector

- modifies the coordinates 0 to k
- reaches an element with the smallest possible coordinate k

Examples in 2-dim



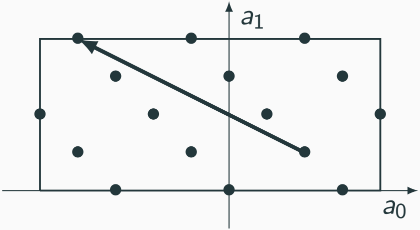
not a 0-transition vector

Using a k-transition vector

- modifies the coordinates 0 to k
- reaches an element with the smallest possible coordinate k

Transition-vector

Examples in 2-dim

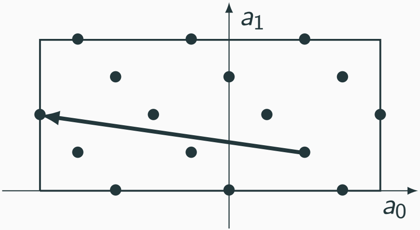


not a 1-transition vector

Using a k-transition vector

- modifies the coordinates 0 to k
- reaches an element with the smallest possible coordinate k

Examples in 2-dim

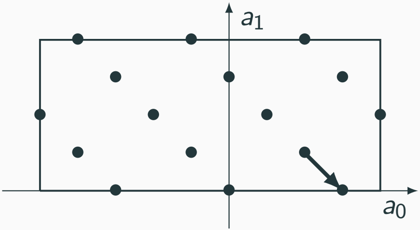


1-transition vector

Using a k-transition vector

- modifies the coordinates 0 to k
- reaches an element with the smallest possible coordinate k

Examples in 2-dim



not a 1-transition vector

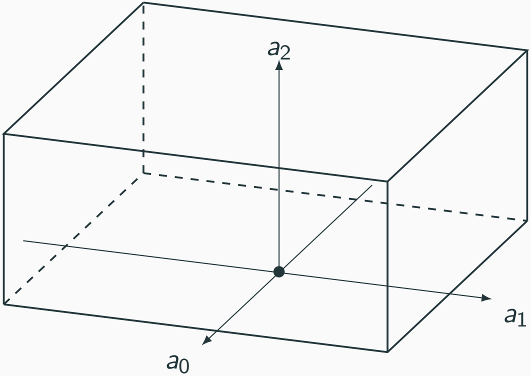
Using a k-transition vector

- modifies the coordinates 0 to k
- reaches an element with the smallest possible coordinate k

Generic sieve algorithms

With oracle

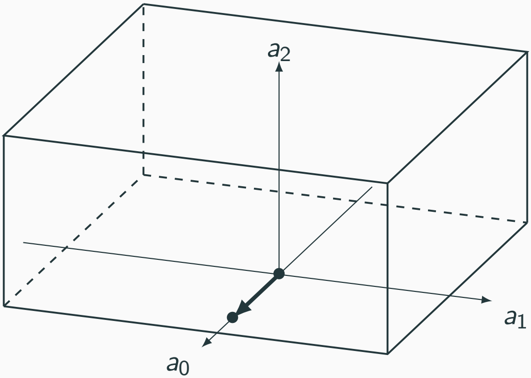
High density



0-TV 1-TV 2-TV

$$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

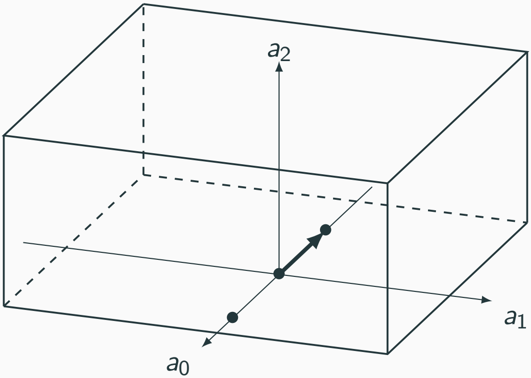
High density



$$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

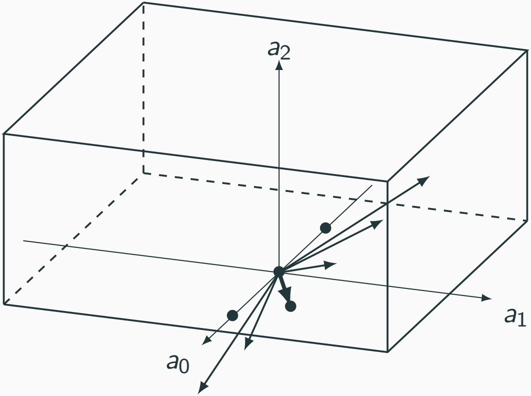
0-TV	1-TV	2-TV
$(3,0,0)$		

High density



	0-TV	1-TV	2-TV
$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$	$(3,0,0)$		

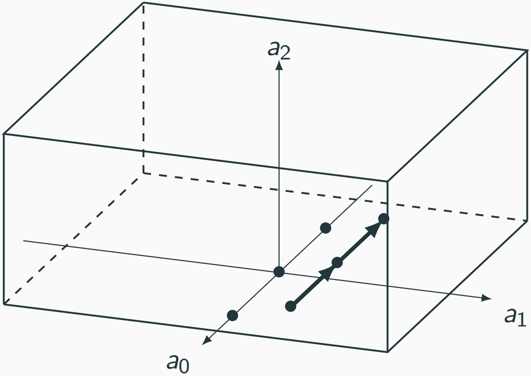
High density



$$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

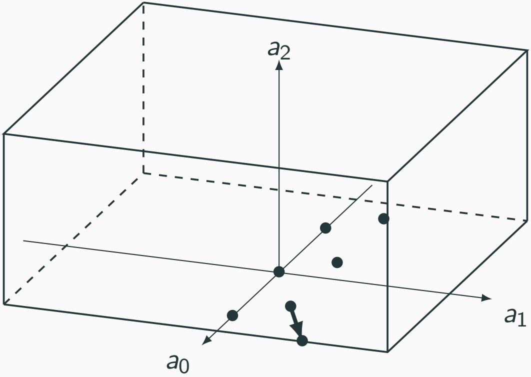
0-TV	1-TV	2-TV
$(3,0,0)$	$(8,1,0)$	
	$(5,1,0)$	
	$(2,1,0)$	
	...	

High density



$$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

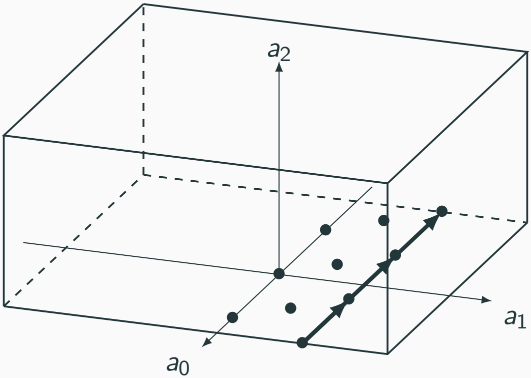
0-TV	1-TV	2-TV
$(3,0,0)$	$(8,1,0)$	
	$(5,1,0)$	
	$(2,1,0)$	
	$\dots$	



$$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

0-TV	1-TV	2-TV
$(3,0,0)$	$(8,1,0)$	
	$(5,1,0)$	
	$(2,1,0)$	
	$\dots$	

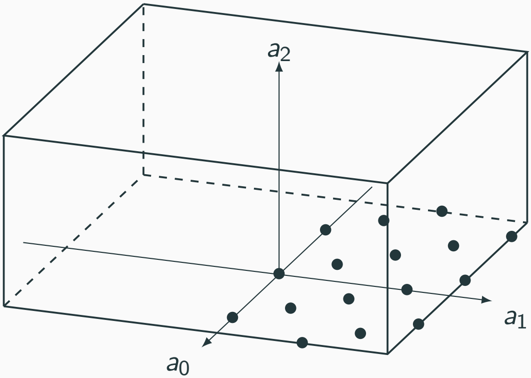
High density



$$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

0-TV	1-TV	2-TV
$(3,0,0)$	$(8,1,0)$	
	$(5,1,0)$	
	$(2,1,0)$	
	$\dots$	

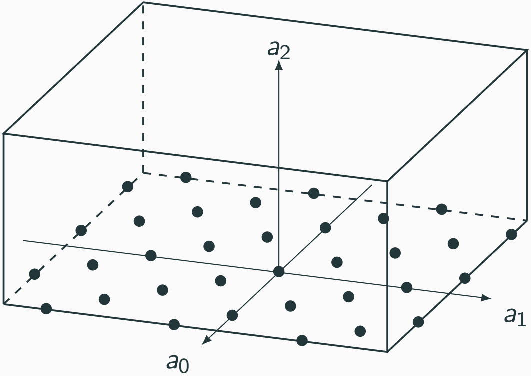
High density



$$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

0-TV	1-TV	2-TV
$(3,0,0)$	$(8,1,0)$	
	$(5,1,0)$	
	$(2,1,0)$	
	$\dots$	

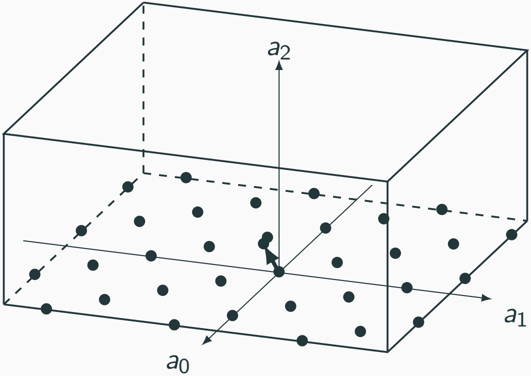
High density



$$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

0-TV	1-TV	2-TV
$(3,0,0)$	$(8,1,0)$	
	$(5,1,0)$	
	$(2,1,0)$	
	$\dots$	

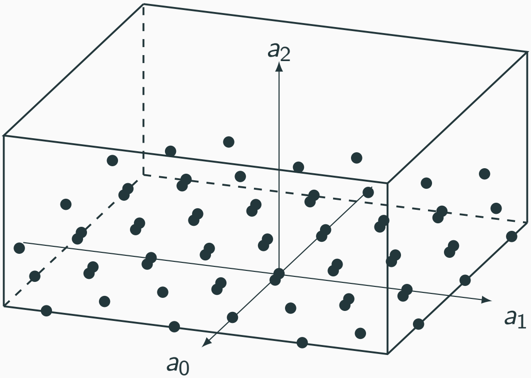
High density



$$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

0-TV	1-TV	2-TV
$(3, 0, 0)$	$(8, 1, 0)$	$(-8, -9, 1)$
	$(5, 1, 0)$	$(1, 0, 1)$
	$(2, 1, 0)$	$(7, 3, 1)$
	$\dots$	$\dots$

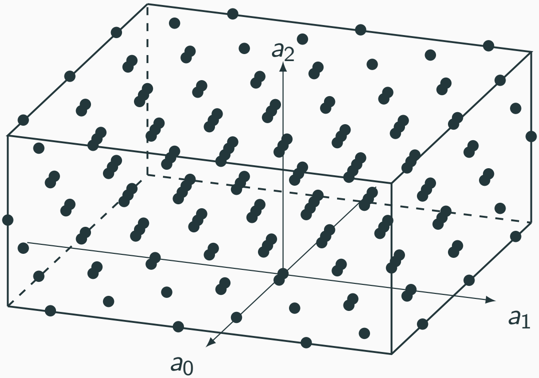
High density



$$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

0-TV	1-TV	2-TV
$(3,0,0)$	$(8,1,0)$	$(-8,-9,1)$
	$(5,1,0)$	$(1,0,1)$
	$(2,1,0)$	$(7,3,1)$
	$\dots$	$\dots$

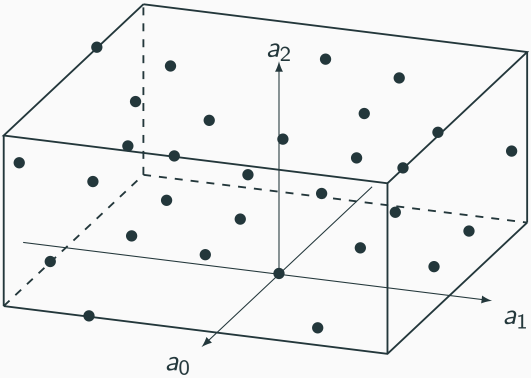
High density



$$\begin{pmatrix} 3 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

0-TV	1-TV	2-TV
$(3, 0, 0)$	$(8, 1, 0)$	$(-8, -9, 1)$
	$(5, 1, 0)$	$(1, 0, 1)$
	$(2, 1, 0)$	$(7, 3, 1)$
	$\dots$	$\dots$

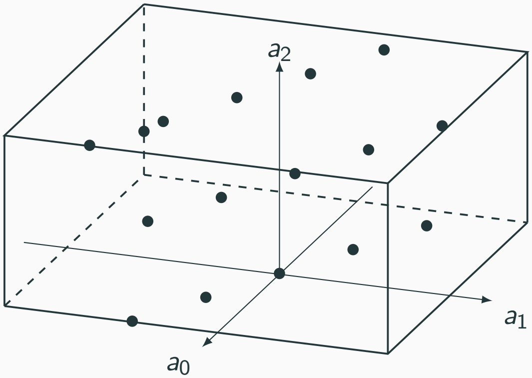
Medium density



$$\begin{pmatrix} 17 & 0 & 0 \\ 10 & 1 & 0 \\ 12 & 0 & 1 \end{pmatrix}$$

0-TV	1-TV	2-TV
$\emptyset$	$(-7, 1, 0)$	$(-2, 2, 1)$
	$(3, 2, 0)$	$(-5, 0, 1)$
	$(-4, 3, 0)$	$(2, -1, 1)$
		$\dots$

Low density



$$\begin{pmatrix} 29 & 0 & 0 \\ 27 & 1 & 0 \\ 14 & 0 & 1 \end{pmatrix}$$

0-TV	1-TV	2-TV
$\emptyset$	$(-2, 1, 0)$	$(-5, -5, 1)$
		$(-1, -7, 1)$
		$(-1, 0, 2)$
		...

Disappointing reality (back to -XII)

Efficient oracles do not exist

- line sieve
- plane sieve

Solution

Introduce a weaker notion than the transition-vectors

Generic sieve algorithms

Without oracle

Nearly-transition-vector

Using a k–transition-vector

- modifies the coordinates 0 to k
- reaches an element with **the smallest possible** coordinate k

Using a k–nearly-transition-vector

- modifies the coordinates 0 to k
- reaches an element with **a different** coordinate k

Shape of the nearly-transition-vectors

Mimic the transition-vectors

- lattice of volume r
- $\mathcal{S} = \underbrace{[S_0^m, S_0^M)}_{\text{length}=l_0} \times \underbrace{[S_1^m, S_1^M)}_{\text{length}=l_1} \times \cdots \times \underbrace{[S_{t-1}^m S_{t-1}^M)}_{\text{length}=l_{t-1}}$
- $\ell < t$ represents the density of the lattice in $\mathcal{S}$

Shape of transition-vectors in 3-dim

- high ($\ell = 0$): $(r, 1, 1)$
- medium ($\ell = 1$): $(l_0, r/l_0, 1)$
- low ($\ell = 2$): $(l_0, l_1, r/(l_0 l_1))$

Shape

$(l_0, l_1, \ldots, l_{\ell-1}, r/(l_0 \times l_1 \times \cdots \times l_{\ell-1}), 1, 1, \ldots, 1)$

Find (some) nearly-transition-vectors

Steps

1. Skew basis reduction
2. Small linear combinations

Skew-small-vectors are all the produced vectors

Patterns of the skew-small-vectors ($t = 6$ and $\ell = 2$)

k	`globalntv`	`localntv`	`sparsentv`
0	$(>0,0,0,0,0,0)$	$(>0,0,0,0,0,0)$	$(>0,0,0,0,0,0)$
1	$(\cdot,>0,0,0,0,0)$	$(\cdot,>0,0,0,0,0)$	$(\cdot,>0,0,0,0,0)$
2	$(\cdot,\cdot,>0,0,0,0)$	$(\cdot,\cdot,>0,0,0,0)$	$(\cdot,\cdot,>0,0,0,0)$
3	$(\cdot,\cdot,\cdot,>0,0,0)$	$(\cdot,\cdot,\cdot,1,0,0)$	$(\cdot,\cdot,\cdot,1,0,0)$
4	$(\cdot,\cdot,\cdot,\cdot,>0,0)$	$(\cdot,\cdot,\cdot,\cdot,1,0)$	$(\cdot,\cdot,\cdot,0,1,0)$
5	$(\cdot,\cdot,\cdot,\cdot,\cdot,>0)$	$(\cdot,\cdot,\cdot,\cdot,\cdot,1)$	$(\cdot,\cdot,\cdot,0,0,1)$

Sieve algorithm

Inputs: lattice Λ, $k < t$, a cuboid $\mathcal{S}$ and TV's

Outputs: elements in $\Lambda \cap \mathcal{S}$

0. set $\boldsymbol{a}$ to $\mathbf{0}$
1. while $\boldsymbol{a}[k] < S_k^M$
   1.1 report $\boldsymbol{a}$
   1.2 if $k > 0$, call recursively the procedure with $\boldsymbol{a}$ and $k - 1$
   1.3 add $\boldsymbol{v}$ to $\boldsymbol{a}$, $\boldsymbol{v}$ a k-TV and $(\boldsymbol{a} + \boldsymbol{v})[k]$ the smallest possible
2. recover initial $\boldsymbol{a}$
3. while $\boldsymbol{a}[k] \geq S_k^m$
   3.1 report $\boldsymbol{a}$
   3.2 as 1.2 and 1.3 with $\boldsymbol{a} - \boldsymbol{v}$

Sieve algorithm

Inputs: lattice Λ, $k < t$, a cuboid $\mathcal{S}$, *tv*'s and *ssv*'s

Outputs: elements in $\Lambda \cap \mathcal{S}$

0. set $\boldsymbol{a}$ to $\mathbf{0}$
1. while $\boldsymbol{a}[k] < S_k^M$
   - 1.1 report $\boldsymbol{a}$
   - 1.2 if $k > 0$, call recursively the procedure with $\boldsymbol{a}$ and $k-1$
   - 1.3 add $\boldsymbol{v}$ to $\boldsymbol{a}$, $\boldsymbol{v}$ a k-*tv* and $(\boldsymbol{a}+\boldsymbol{v})[k]$ the smallest possible
   - 1.4 if no $\boldsymbol{a}+\boldsymbol{v}$ in $\mathcal{S}$, find new $\boldsymbol{a}$ using k–skew-small-vector
2. recover initial $\boldsymbol{a}$
3. while $\boldsymbol{a}[k] \geq S_k^m$
   - 3.1 report $\boldsymbol{a}$
   - 3.2 as 1.2, 1.3 and 1.4 with $\boldsymbol{a}-\boldsymbol{v}$

Common fall-back strategy

Algorithm (for 1.4)

1. while $\boldsymbol{a}[k] < S_k^M$
    1.1 for all k–skew-small-vectors $\boldsymbol{v}$
        1.1.1 look for $\boldsymbol{e}$ that reduces some coefficients of $\boldsymbol{a} + \boldsymbol{v}$
        1.1.2 if $\boldsymbol{a} + \boldsymbol{v} - \boldsymbol{e}$ in $\mathcal{S}$, return $\boldsymbol{a} + \boldsymbol{v} - \boldsymbol{e}$
    1.2 set $\boldsymbol{a}$ to one of the vector $\boldsymbol{a} + \boldsymbol{v} - \boldsymbol{e}$
2. return `fail`

Very costly, parsimoniously used

Generic sieve algorithms

Practical results

Practical results

	4-dim	6-dim
Accuracy	fair	
# *ssv*'s	fair	prohibitive
# fb's	fair	prohibitive

Running time in 4-dim

- Low density
  - new sieves much faster than the plane sieve
- Very low density
  - `localntv` for low density faster than `globalntv`

Conclusion and perspectives

Conclusion

Three new sieve algorithms

- `globalntv`, `localntv` and `sparsentv`
- implemented in `ntv.sage`, available in [CADO-NFS]

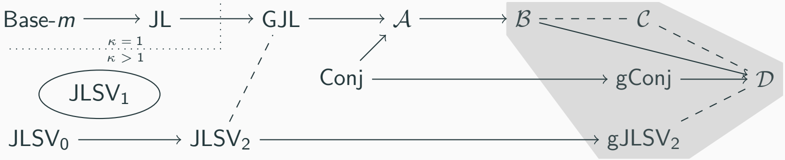
Practical results

- 4-dim reachable
- 6-dim and beyond unreachable

Challenges for exTNFS

1. Polynomial selection
   - 11 different polynomial selection algorithms
2. Relation collection
   - 4-dim relation collection implementation?
   - cuboid search space?
3. Linear algebra
4. Individual logarithm

Polynomial selections



Bibliography (I)

BaGaGuMo'15a R. Barbulescu, P. Gaudry, A. Guillevic, and F. Morain, *Improving NFS for the discrete logarithm problem in non-prime finite fields*, EUROCRYPT 2015.

BaGaGuMo'15b R. Barbulescu, P. Gaudry, A. Guillevic and F. Morain, Announced in *Individual Discrete Logarithm in* $GF(p^k)$, http://www.lix.polytechnique.fr/~guillevic/docs/guillevic-catrel15-talk.pdf.

CADO-NFS The CADO-NFS Development Team, *CADO-NFS, an implementation of the number field sieve algorithm*, 2018, http://cado-nfs.gforge.inria.fr/.

DLDB L. Grémy and A. Guillevic, *DiscreteLogDB, a database of computations of discrete logarithms*, 2018, https://gitlab.inria.fr/dldb/discretelogdb.

Bibliography (II)

FrKl'05 J. Franke and T. Kleinjung, *Continued fractions and lattice sieving*, SHARCS 2005.

GaGrVi'16 P. Gaudry, L. Grémy, and M. Videau, *Collecting relations for the number field sieve in* $GF(p^6)$, LMS J. Comput. Math 19 (2016).

GaGuMo'15 P. Gaudry, A. Guillevic and F. Morain, *Discrete logarithm record in GF(p^3) of 592 bits (180 decimal digits)*, NMBRTHRY Archives, 2015.

GrGuMoTh'18 L. Grémy, A. Guillevic, F. Morain, and E. Thomé, *Computing discrete logarithms in* $GF(p^6)$, SAC 2017.

GrGuMo'17 L. Grémy, A. Guillevic, F. Morain, *Breaking DLP in* $GF(p^5)$ *using 3-dimensional sieving*, 2017.

HaAoKoTa'13 K. Hayasaka, K. Aoki, T. Kobayashi, and T. Takagi, *An Experiment of Number Field Sieve for Discrete Logarithm Problem over* $GF(p^{12})$, Number Theory and Cryptography.

Bibliography (III)

HaAoKoTa'15 K. Hayasaka, K. Aoki, T. Kobayashi, and T. Takagi, *A construction of 3-dimensional lattice sieve for number field sieve over* $\mathbb{F}_{p^n}$, ePrint archive 2015/1179.

KiBa'16 T. Kim and R. Barbulescu, *Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case*, CRYPTO 2016.

KlDiLePrSt'17 T. Kleinjung, C. Diem, A. Lenstra, C. Priplata, and C. Stahlke, *Computation of a 768-Bit Prime Field Discrete Logarithm*, EUROCRYPT 2017.

JoLeSmVe'06 A. Joux, R. Lercier, N. Smart, and F. Vercauteren, *The Number Field Sieve in the Medium Prime Case*, CRYPTO 2006.

Schirokauer'92 Schirokauer, O. A.: On pro-finite groups and on discrete logarithms. Ph.D. thesis, 1992.

Zajac'08 P. Zajac, *Discrete logarithm problem in degree six finite fields*, Ph.D. thesis, 2008.

Sources

- https://fr.wikipedia.org/wiki/Alice_(Modigliani)
- https://en.wikipedia.org/wiki/Robert_Schumann