CSE 291-E: Applied Cryptography

Nadia Heninger

UCSD

Fall 2020 Lecture 17

Legal Notice

The Zoom session for this class will be recorded and made available asynchronously on Canvas to registered students.

Announcements

1. HW 7 is due today!

2. HW 8 is out today! Due in 1.5 weeks!

Last time:

- Number field sieve

This time:

- Lattice-based cryptanalysis

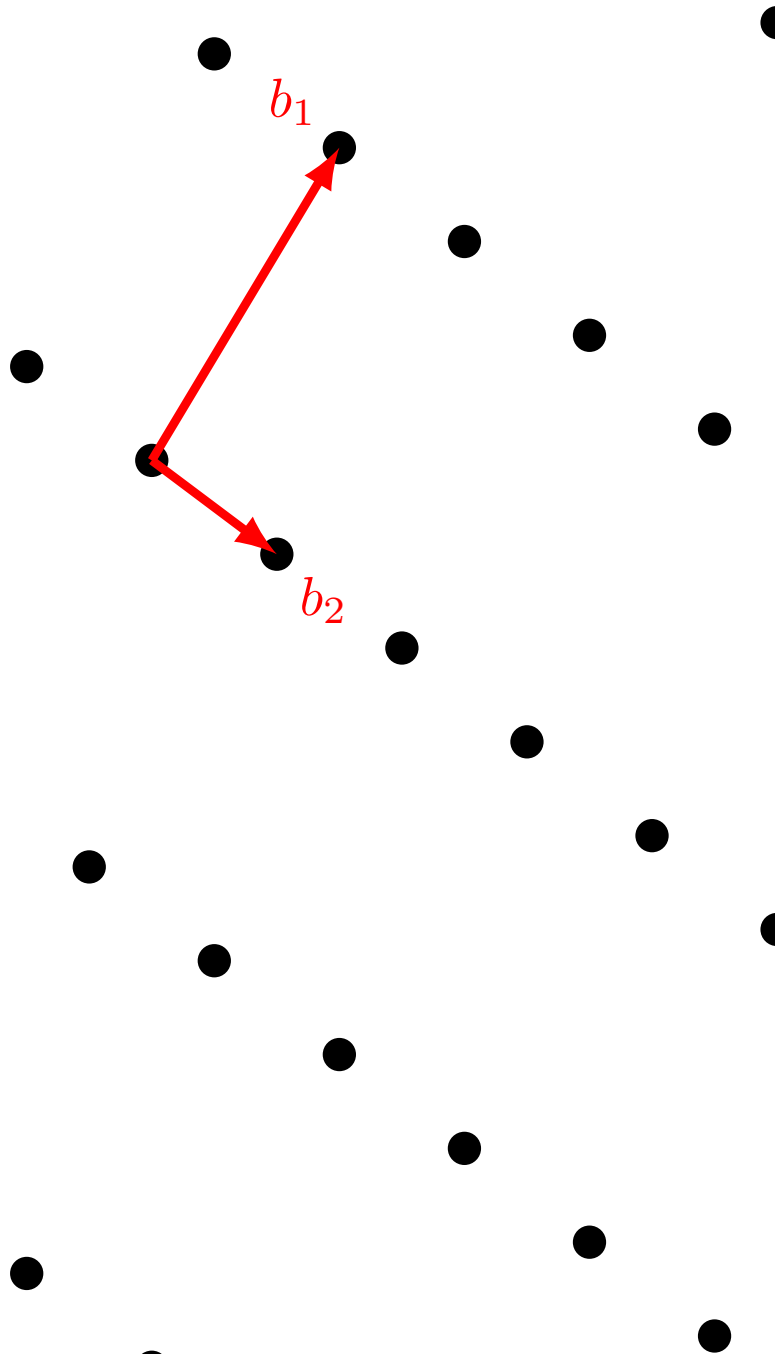
What is a lattice?

Definition

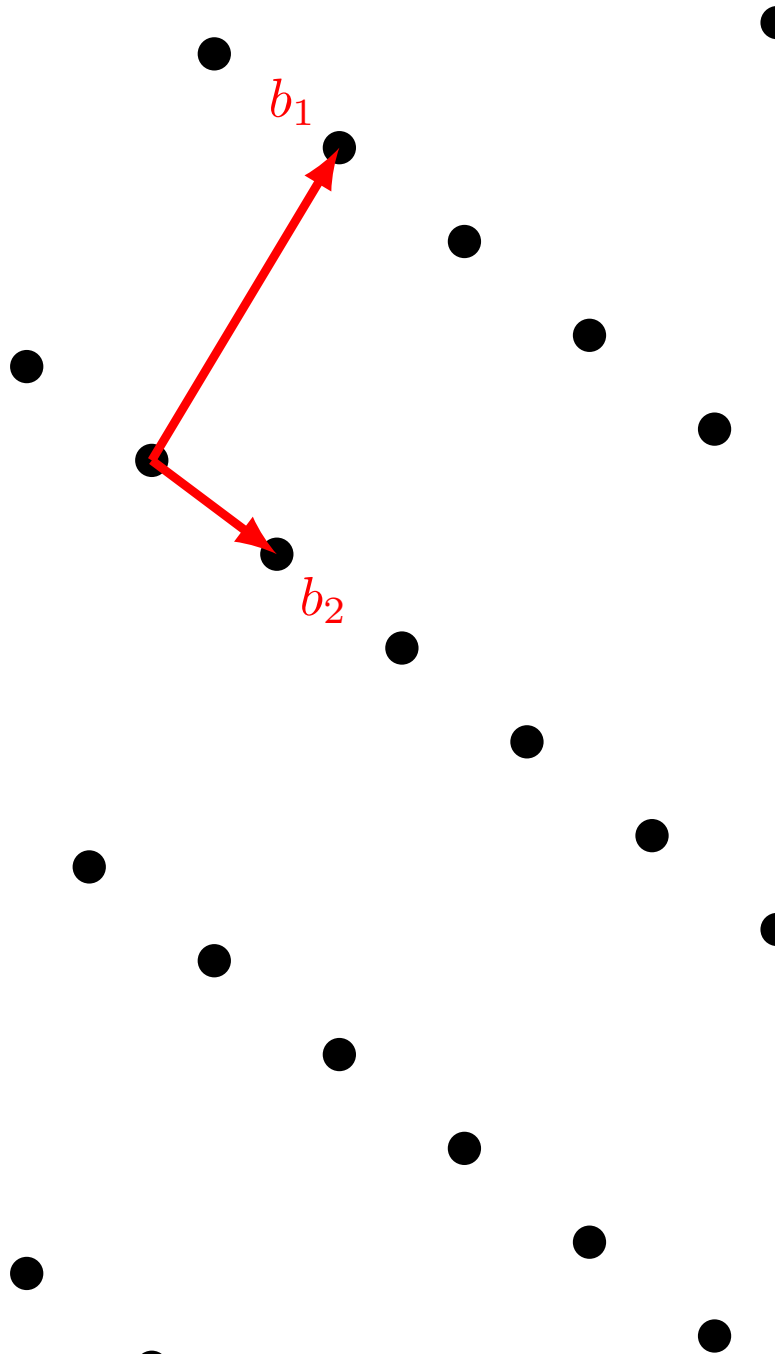
A **lattice** is a subset of $\mathbb{R}^n$ generated by integer linear combinations of some linearly independent basis $\{b_1, \ldots, b_n\}$.

Can represent as Cartesian coordinates: origin $(0, 0, \ldots, 0)$ $b_i = (z_1, \ldots, z_n)$.

- Has algebraic properties (it's a group under addition).
- Has geometric properties (it lives in $\mathbb{R}^n$ so has dot product, distance).



What is a lattice?



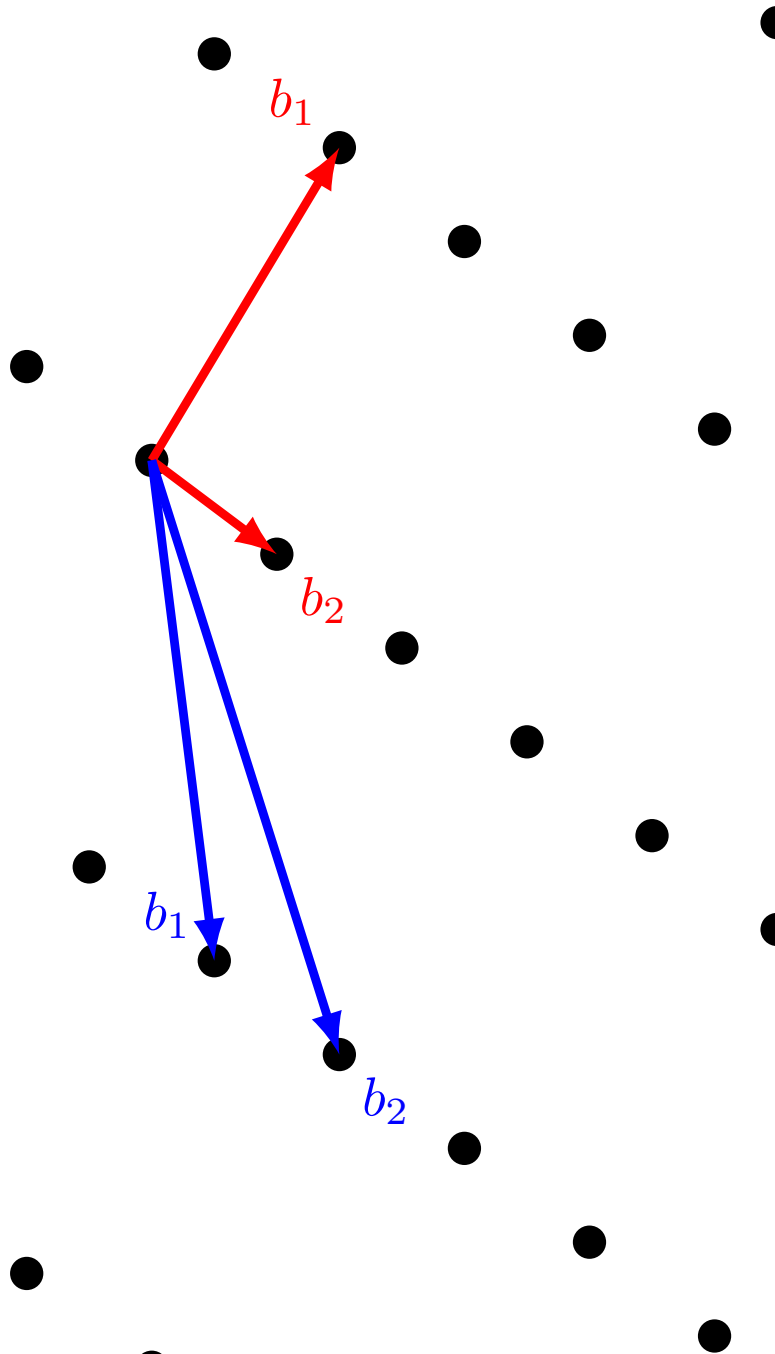
Definition

A **lattice** is a discrete additive subgroup of $\mathbb{R}^n$.

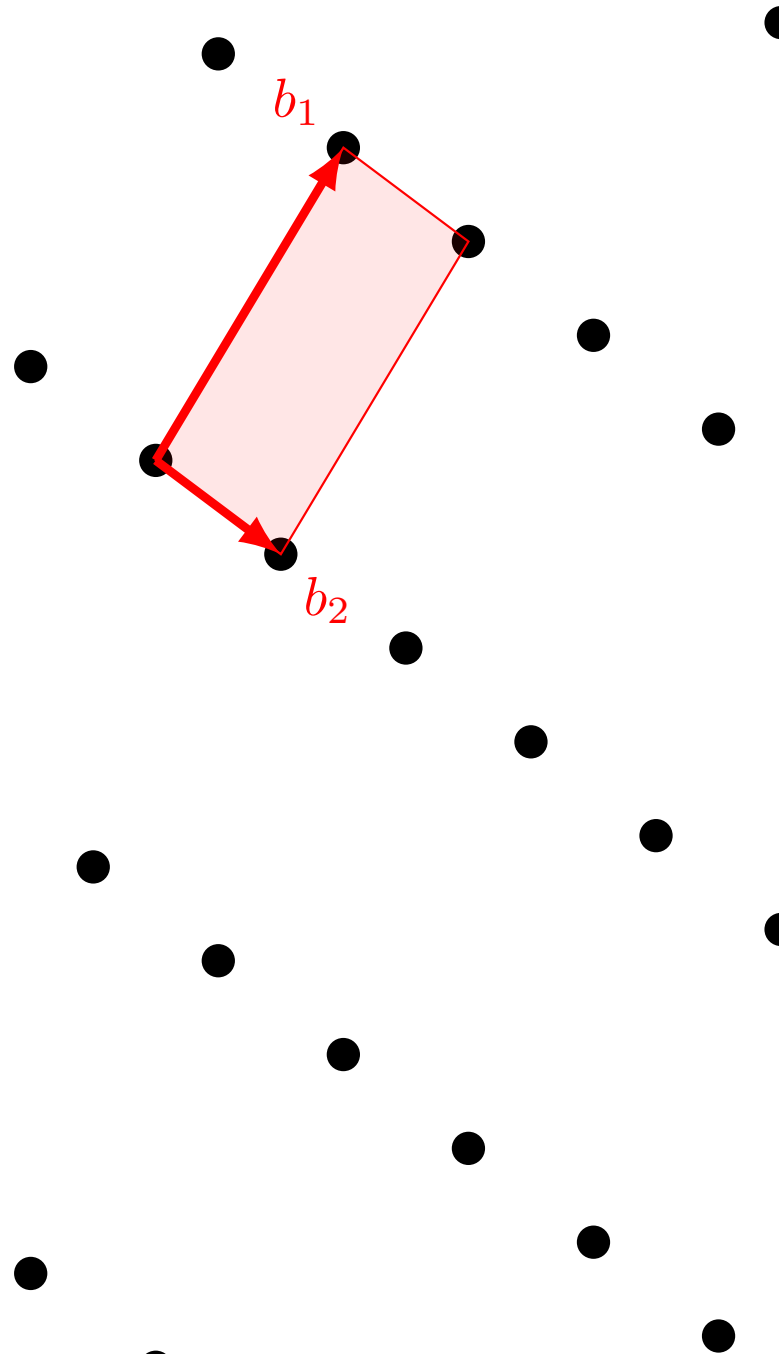
- Discrete: $\exists \delta > 0$ s.t. $|v_i - v_j| > \delta$ $\forall\ v_i, v_j \in L(B)$.
- Additive subgroup: closed under addition.

Properties of lattices: Bases

- In n dimensions a lattice has a basis of size at most n.
- The basis is not unique.
- Let $L(B)$ be the lattice generated by basis B. Deciding if $L(B) = L(B')$ for $B \neq B'$ is efficient. The Hermite Normal Form is unique and efficient to compute. Check if HNF(B) = HNF(B').



Properties of lattices: Determinant



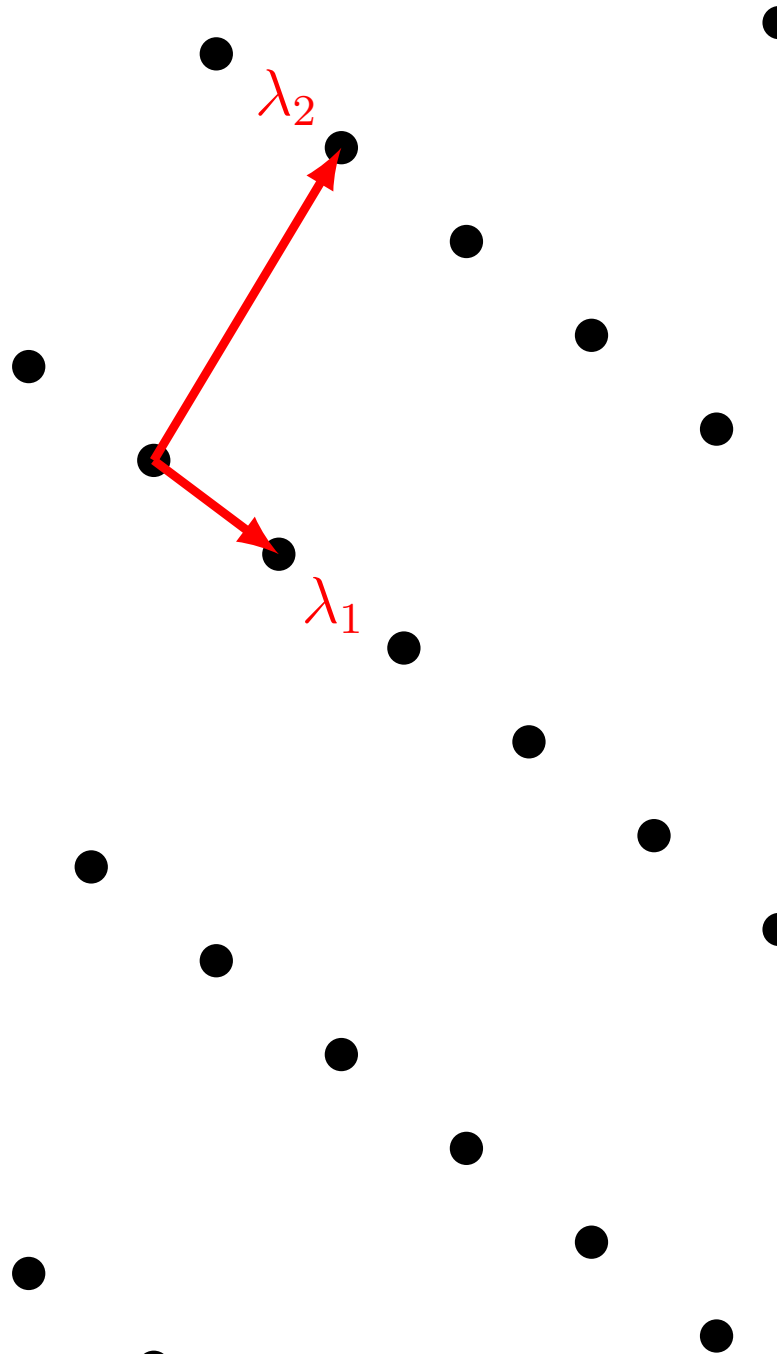
Definition

The **determinant** of a lattice with a basis matrix B is $|\det B|$.

- The determinant is invariant for a given lattice.
- Gives volume of fundamental parallelepiped.

Properties of lattices: Minima

Let $\lambda_1 > 0$ be the length of the shortest vector in the lattice.



Theorem (Minkowski)

$\lambda_1(L) < \sqrt{n} \det L^{1/n}$

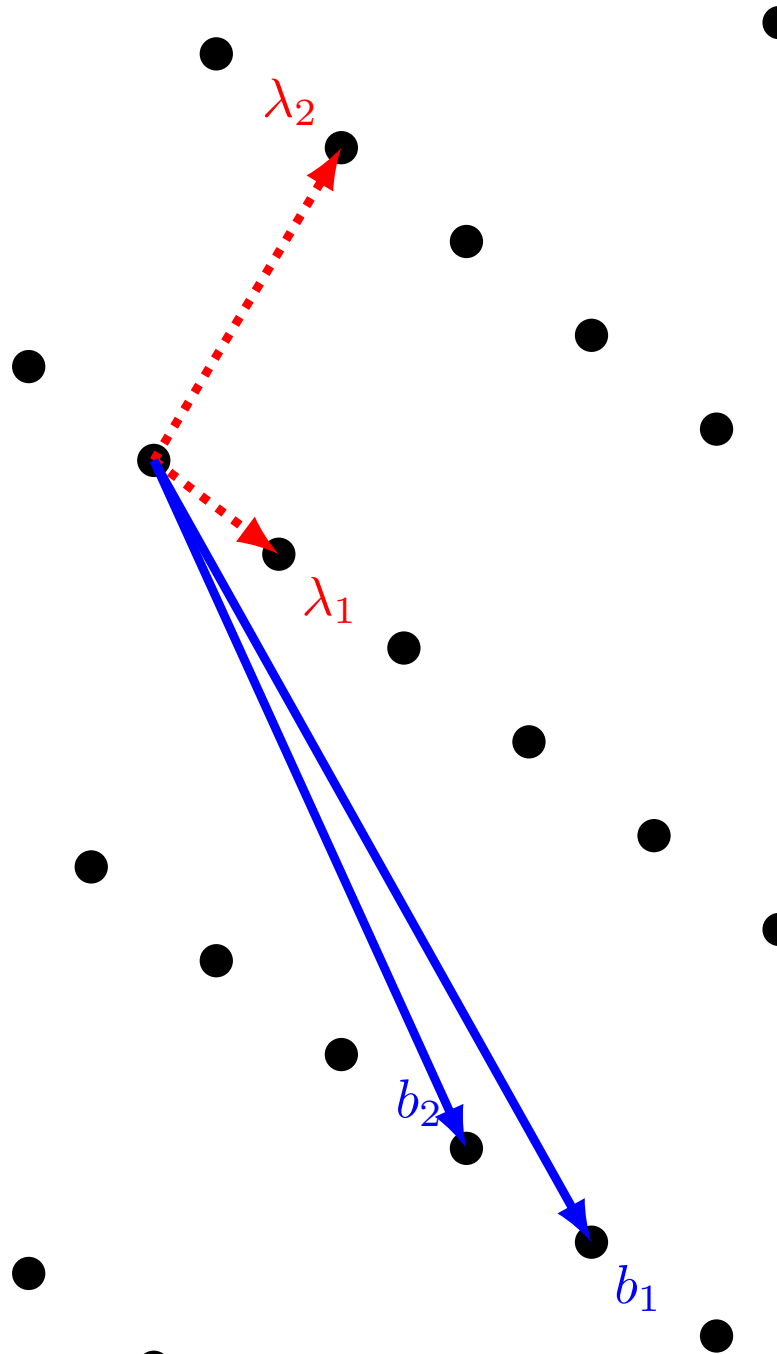
Can define *successive minima* λ_i, the length of the shortest vector linearly independent to the vectors achieving the $i-1$ successive minima.

Computational problems on lattices: SVP

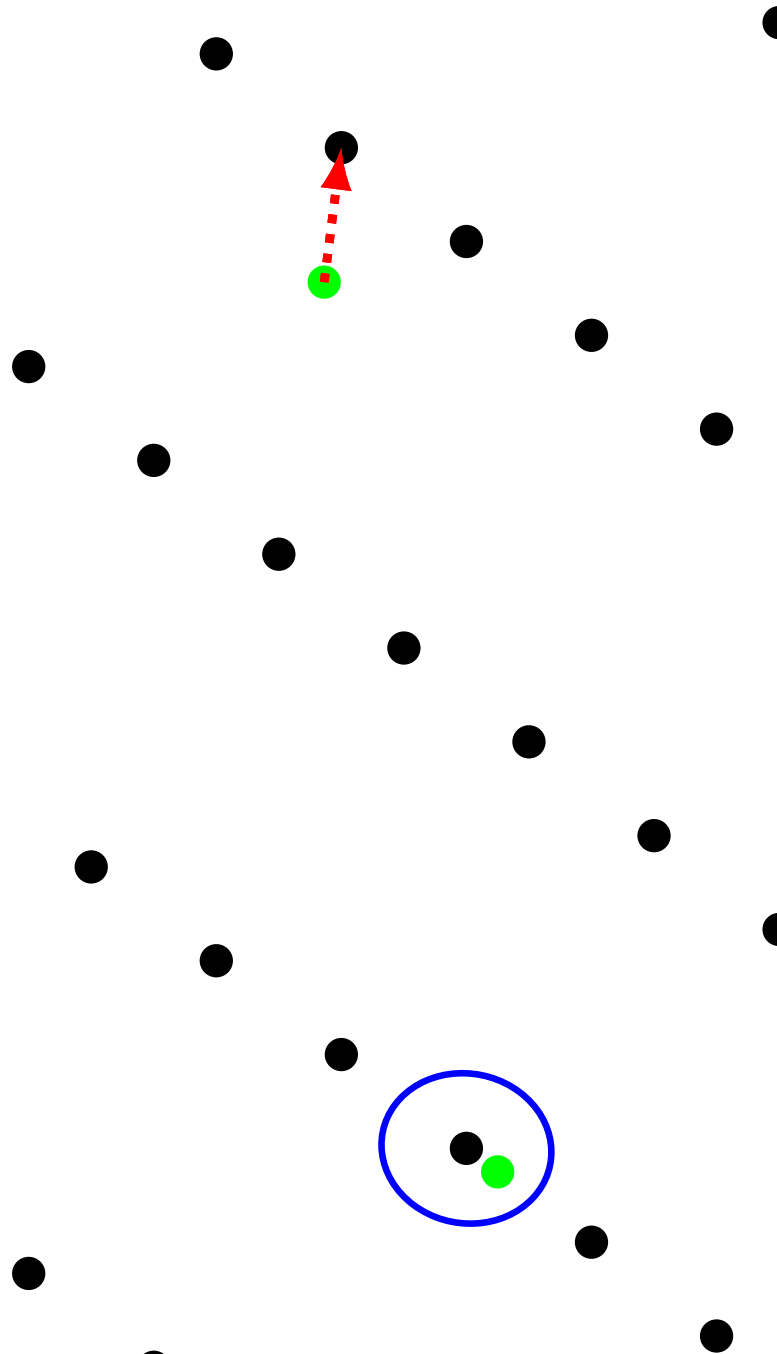
Shortest Vector Problem (SVP)

Given an arbitrary basis for L, find the shortest vector in L.

- SVP is NP-hard.



Computational problems on lattices: CVP



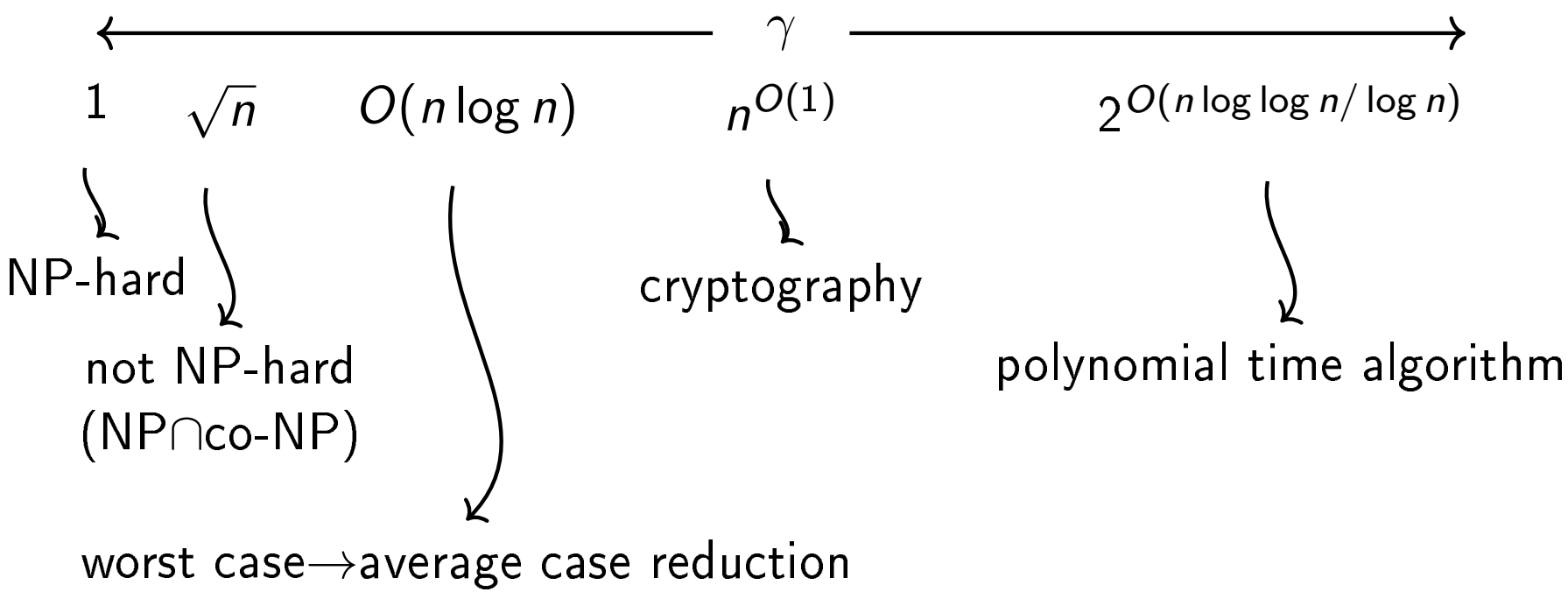
Closest Vector Problem (CVP)

Given an arbitrary basis for L, and a point x find the vector in L closest to x.

- CVP is NP-hard.

Approximation results for SVP

Input: Lattice basis B.
Desired output: Vector of length $\gamma\lambda_1(L(B))$.



Algorithmic results for SVP

Lenstra Lenstra Lovasz (LLL)

Given a basis for a lattice can in polynomial time find a *reduced* basis $\{b_i\}$ s.t.

$$|b_i| \leq 2^{(n-1)/2}\lambda_i$$

Algorithmic results for SVP

Lenstra Lenstra Lovasz (LLL)

Given a basis for a lattice can in polynomial time find a *reduced* basis $\{b_i\}$ s.t.

$$|b_i| \leq 2^{(n-1)/2}\lambda_i$$

Theorem (LLL (Simplified Version))

We can find a vector of length

$$|v| < 2^{\dim L}(\det L)^{1/\dim L}$$

Algorithmic results for SVP

Lenstra Lenstra Lovasz (LLL)

Given a basis for a lattice can in polynomial time find a *reduced* basis $\{b_i\}$ s.t.

$$|b_i| \leq 2^{(n-1)/2}\lambda_i$$

Theorem (LLL (Simplified Version))

We can find a vector of length

$$|v| < 2^{\dim L}(\det L)^{1/\dim L}$$

- In practice on random lattices, LLL finds $v = 1.02^n(\det L)^{1/\dim L}$. [Nguyen,Stehle]

Algorithmic results for SVP

Lenstra Lenstra Lovasz (LLL)

Given a basis for a lattice can in polynomial time find a *reduced* basis $\{b_i\}$ s.t.

$$|b_i| \leq 2^{(n-1)/2}\lambda_i$$

Theorem (LLL (Simplified Version))

We can find a vector of length

$$|v| < 2^{\dim L}(\det L)^{1/\dim L}$$

- In practice on random lattices, LLL finds $v = 1.02^n(\det L)^{1/\dim L}$. [Nguyen,Stehle]

BKZ

Given a lattice basis, can in time $2^{O(k)}$ find a reduced basis s.t. $|b_i| \leq k^{O(n/k)}\lambda_i$.

The "two faces" of lattices in cryptography

- **Cryptanalysis:** Can use approximation algorithms for SVP in lattices to cryptanalyze a wide variety of classical cryptography:
  - Attacks on low public exponent RSA
  - Factoring with partial knowledge
  - (EC)DSA with partial information about nonces
  - Knapsack-based cryptosystems

- **Cryptographic constructions:**
  - Lattice problems appear to be hard to solve for quantum computers, so lattice-based cryptosystems among most promising candidates for post-quantum cryptography.
  - Algebraic structure of lattices leads to many interesting cryptographic constructions that may someday be practical, like fully homomorphic encryption, identity-based encryption, etc.

History: Lattices and cryptography

1910	Minkowski's geometry of numbers
1973/1977	Public-key cryptography invented (GCHQ/RSA)
1978	Knapsack cryptography invented (Merkle-Hellmann)
1982	CVP NP-hard (van Emde Boas)
1982	LLL lattice basis reduction algorithm (Lenstra-Lenstra-Lovasz)
1983	LLL algorithm used against knapsack cryptosystems (Lagarias-Odlyzko)
1996	Lattice-based cryptosystems invented (Ajtai-Dwork)
1997	SVP NP-hard (Ajtai)
2005	LWE problem (Regev)
2009	Fully homomorphic encryption using ideal lattices (Gentry)

Historical Interlude: Subset Sum

Subset Sum Problem

Input: Integers $a_1, \ldots, a_n$, target integer T.
Goal: Find a subset $\sum_S a_i = T$.

- NP-hard
- First attempt to base cryptography off of NP-hardness.
- All schemes have a “trapdoor" that lets the decrypter solve the problem. (e.g. super-increasing sequence working modulo some number.)

Solving subset sum with lattices

Input: Integers $a_1, \ldots, a_n$, target integer T.

Generate lattice from rows of matrix

$$\begin{bmatrix} 1 & & & a_1 \\ & 1 & & a_2 \\ & & \ddots & \vdots \\ & & & -T \end{bmatrix}$$

Solving subset sum with lattices

Input: Integers $a_1, \ldots, a_n$, target integer T.

Generate lattice from rows of matrix

$$\begin{bmatrix} 1 & & & a_1 \\ & 1 & & a_2 \\ & & \ddots & \vdots \\ & & & -T \end{bmatrix}$$

A solution $\sum_i b_i a_i = T$ $b_i \in \{0, 1\}$ determines a vector

$$v = (b_1, b_2, \ldots, 0) \qquad |v|_2 \approx \sqrt{n/2}$$

Solving subset sum with lattices

Input: Integers $a_1, \ldots, a_n$, target integer T.

Generate lattice from rows of matrix

$$\begin{bmatrix} 1 & & & a_1 \\ & 1 & & a_2 \\ & & \ddots & \vdots \\ & & & -T \end{bmatrix}$$

A solution $\sum_i b_i a_i = T$ $b_i \in \{0, 1\}$ determines a vector

$$v = (b_1, b_2, \ldots, 0) \qquad |v|_2 \approx \sqrt{n/2}$$

- $\det L = T$, $\dim L = n + 1$; expect random non-solution vectors to have size $\sqrt{n} \det L^{1/\dim L}$; LLL has approximation factor 1.02^n.
- LLL or BKZ might find short v when $|v| = \sqrt{n/2} < T^{1/(n+1)}$
- Proposed knapsack cryptosystems contained "trapdoors" that made problem easier to solve.

What's wrong with this RSA example?

```
message = Integer('squeamishossifrage',base=35)
N = random_prime(2^512)*random_prime(2^512)
c = message^3 % N
```

What's wrong with this RSA example?

```
message = Integer('squeamishossifrage',base=35)
N = random_prime(2^512)*random_prime(2^512)
c = message^3 % N

sage: Integer(c^(1/3)).str(base=35)
'squeamishossifrage'
```

What's wrong with this RSA example?

```
message = Integer('squeamishossifrage',base=35)
N = random_prime(2^512)*random_prime(2^512)
c = message^3 % N

sage: Integer(c^(1/3)).str(base=35)
'squeamishossifrage'
```

The message is too small.

This is why we use padding.

```
N = random_prime(2^150)*random_prime(2^150)
message = Integer('thepasswordfortodayisswordfish',base=35)
c = message^3 % N
```

```
N = random_prime(2^150)*random_prime(2^150)
message = Integer('thepasswordfortodayisswordfish',base=35)
c = message^3 % N

sage: int(c^(1/3))==message
False
```

```
N = random_prime(2^150)*random_prime(2^150)
message = Integer('thepasswordfortodayisswordfish',base=35)
c = message^3 % N
```

This is a stereotyped message. We might be able to guess the format.

```
N = random_prime(2^150)*random_prime(2^150)
message = Integer('thepasswordfortodayisswordfish',base=35)
c = message^3 % N


a = Integer('thepasswordfortodayis000000000',base=35)
```

```
N = random_prime(2^150)*random_prime(2^150)
message = Integer('thepasswordfortodayisswordfish',base=35)
c = message^3 % N


a = Integer('thepasswordfortodayis000000000',base=35)

X = Integer('xxxxxxxxx',base=35)
M = matrix([[X^3, 3*X^2*a, 3*X*a^2, a^3-c],
            [0,N*X^2,0,0],[0,0,N*X,0],[0,0,0,N]])
```

```
N = random_prime(2^150)*random_prime(2^150)
message = Integer('thepasswordfortodayisswordfish',base=35)
c = message^3 % N



a = Integer('thepasswordfortodayis000000000',base=35)

X = Integer('xxxxxxxxx',base=35)
M = matrix([[X^3, 3*X^2*a, 3*X*a^2, a^3-c],
            [0,N*X^2,0,0],[0,0,N*X,0],[0,0,0,N]])

B = M.LLL()
Q = B[0][0]*x^3/X^3+B[0][1]*x^2/X^2+B[0][2]*x/X+B[0][3]
```

```
N = random_prime(2^150)*random_prime(2^150)
message = Integer('thepasswordfortodayisswordfish',base=35)
c = message^3 % N



a = Integer('thepasswordfortodayis000000000',base=35)

X = Integer('xxxxxxxxx',base=35)
M = matrix([[X^3, 3*X^2*a, 3*X*a^2, a^3-c],
            [0,N*X^2,0,0],[0,0,N*X,0],[0,0,0,N]])

B = M.LLL()
Q = B[0][0]*x^3/X^3+B[0][1]*x^2/X^2+B[0][2]*x/X+B[0][3]

sage: Q.roots(ring=ZZ)[0][0].str(base=35)
'swordfish'
```

What's going on here? Coppersmith's method.

Theorem (Coppersmith)

We can efficiently compute up to $1/e$*-fraction of the bits of an RSA-encrypted message with public exponent* e *if we know the rest of the plaintext.*

```
sage: N.nbits()
296
sage: Integer('swordfish',base=35).nbits()
46
```

What's going on here? Coppersmith's method.

Theorem (Coppersmith)

Given a polynomial f of degree d and N, we can efficiently find all roots r_i satisfying

$$f(r_i) \equiv 0 \bmod N$$

when $|r_i| < N^{1/d}$ in time polynomial in $\log N$ and d.

In our case, our input polynomial looks like

$$f(x) = (a + x)^3 - c \equiv 0 \bmod N$$

We are looking for a root $r = \texttt{swordfish}$ satisfying

$$f(r) = (a + \texttt{swordfish})^3 - c \equiv 0 \bmod N$$

Why is this an interesting theorem?

1. A general method to solve polynomials mod N would break RSA: If c is a ciphertext,

$$x^e - c \equiv 0 \bmod N$$

   has a root $x = m$ for m our original message.

2. There is an efficient algorithm to solve equations mod primes.
   - For a composite, factor into primes, solve mod each prime, and use Chinese remainder theorem and Hensel lifting to lift solution mod N.

3. By accepting a bound on solution size, Coppersmith's method lets us solve equations without factoring N.

Coppersmith's Algorithm Outline

Input: polynomial f, modulus N.
Output: a root r modulo N.

In our example, we have $f(x) = (x + a)^3 - c$.

We will construct a new polynomial $Q(x)$ so that

$$Q(r) = 0 \qquad \text{over the integers.}$$

If we construct $Q(x)$ as

$$Q(x) = s(x)f(x) + t(x)N$$

with $s(x), t(x) \in \mathbb{Z}[x]$, then by construction

$$Q(r) \equiv 0 \bmod N$$

(In other words, $Q(x) \in \langle f(x), N \rangle$ over $\mathbb{Z}[x]$.)

Manipulating polynomials

Input: $f(x) = x^3 + f_2x^2 + f_1x + f_0$,N
Output: $Q(x) \in \langle f(x), N \rangle$ over $\mathbb{Z}[x]$.

If we only care about polynomials Q of degree 3, then

$$Q(x) = c_3 f(x) + c_2 N x^2 + c_1 N x + c_0 N$$

with $c_3, c_2, c_1, c_0 \in \mathbb{Z}$.

$$\begin{array}{llllllll}
 & c_3 & (x^3 & + & f_2x^2 & + & f_1x & + & f_0) \\
+ & c_2 & & & Nx^2 & & & & \\
+ & c_1 & & & & & Nx & & \\
+ & c_0 & & & & & & & N \\
\hline
 & & Q_3x^3 & + & Q_2x^2 & + & Q_1x & + & Q_0
\end{array}$$

Manipulating polynomials as coefficient vectors

We can represent elements of $\mathbb{Z}[x]$ as coefficient vectors:

$$g_d x^d + g_{d-1} x^{d-1} + \cdots + g_0 \qquad \leftrightarrow \qquad (g_d, g_{d-1}, \ldots, g_0)$$

If we construct the matrix

$$\begin{bmatrix} 1 & f_2 & f_1 & f_0 \\ & N & & \\ & & N & \\ & & & N \end{bmatrix}$$

Then the coefficient vector representing our polynomial

$$Q(x) = c_3 f(x) + c_2 N x^2 + c_1 N x + c_0 N$$

is an integer combination of the rows of this matrix.

Polynomial coefficient vectors and lattices

The set of vectors generated by integer combinations of the rows of our matrix

$$\begin{bmatrix} 1 & f_2 & f_1 & f_0 \\ & N & & \\ & & N & \\ & & & N \end{bmatrix}$$

is a *lattice*.

Coppersmith's method outline

Input: $f(x) \in \mathbb{Z}[x]$, $N \in \mathbb{Z}$. **Output:** r s.t. $f(r) \equiv 0 \bmod N$.

Intermediate output: $Q(x)$ such that $Q(r) = 0$ over $\mathbb{Z}$.

1. $Q(x) \in \langle f(x), N \rangle$ so $Q(r) \equiv 0 \bmod N$ by construction.

2. If $|r| < R$, then we can bound

$$\begin{aligned} |Q(r)| &= |Q_3 r^3 + Q_2 r^2 + Q_1 r + Q_0| \\ &\leq |Q_3| R^3 + |Q_2| R^2 + |Q_1| R + |Q_0| \end{aligned}$$

3. If $|Q(r)| < N$ and $Q(r) \equiv 0 \bmod N$ then $Q(r) = 0$.

We want a Q in our lattice with short coefficient vector!

Coppersmith's method outline

1. Construct a matrix of coefficient vectors of elements of $\langle f(x), N \rangle$.

2. Run a lattice basis reduction algorithm on this matrix.

3. Construct a polynomial Q from the shortest vector output.

4. Factor Q to find its roots.

Running Coppersmith's method on our example

Input: $f(x) = (x + a)^3 - c$, N
Output: $r < R$ such that $f(r) \equiv 0 \bmod N$.

1. Construct lattice basis

$$\begin{bmatrix} R^3 & 3aR^2 & 3a^2R & a^3 - c \\ & NR^2 & & \\ & & NR & \\ & & & N \end{bmatrix}$$

$\dim L = 4$

$\det L = R^6 N^3$

Factor of R is so that $Q(r) \leq |v|$ for $v \in L$.

Running Coppersmith's method on our example

Input: $f(x) = (x + a)^3 - c$, N
Output: $r < R$ such that $f(r) \equiv 0 \bmod N$.

1. Construct lattice basis

$$\begin{bmatrix} R^3 & 3aR^2 & 3a^2R & a^3 - c \\ & NR^2 & & \\ & & NR & \\ & & & N \end{bmatrix}$$

$\dim L = 4$

$\det L = R^6 N^3$

Factor of R is so that $Q(r) \leq |v|$ for $v \in L$.

2. Ignoring approximation factor, we can solve when

$$|Q(r)| \leq |v_1| \leq \det L^{1/\dim L} < N$$
$$(R^6 N^3)^{1/4} < N$$
$$R < N^{1/6}$$

In my example I chose $\lg N = 296$, $\lg r = 46$.

Achieving the Coppersmith bound $r < N^{1/d}$

1. Generate lattice from subset of $\langle f(x), N \rangle^k$.
2. Allow higher degree polynomials.

Theorem (CHHS 2016)

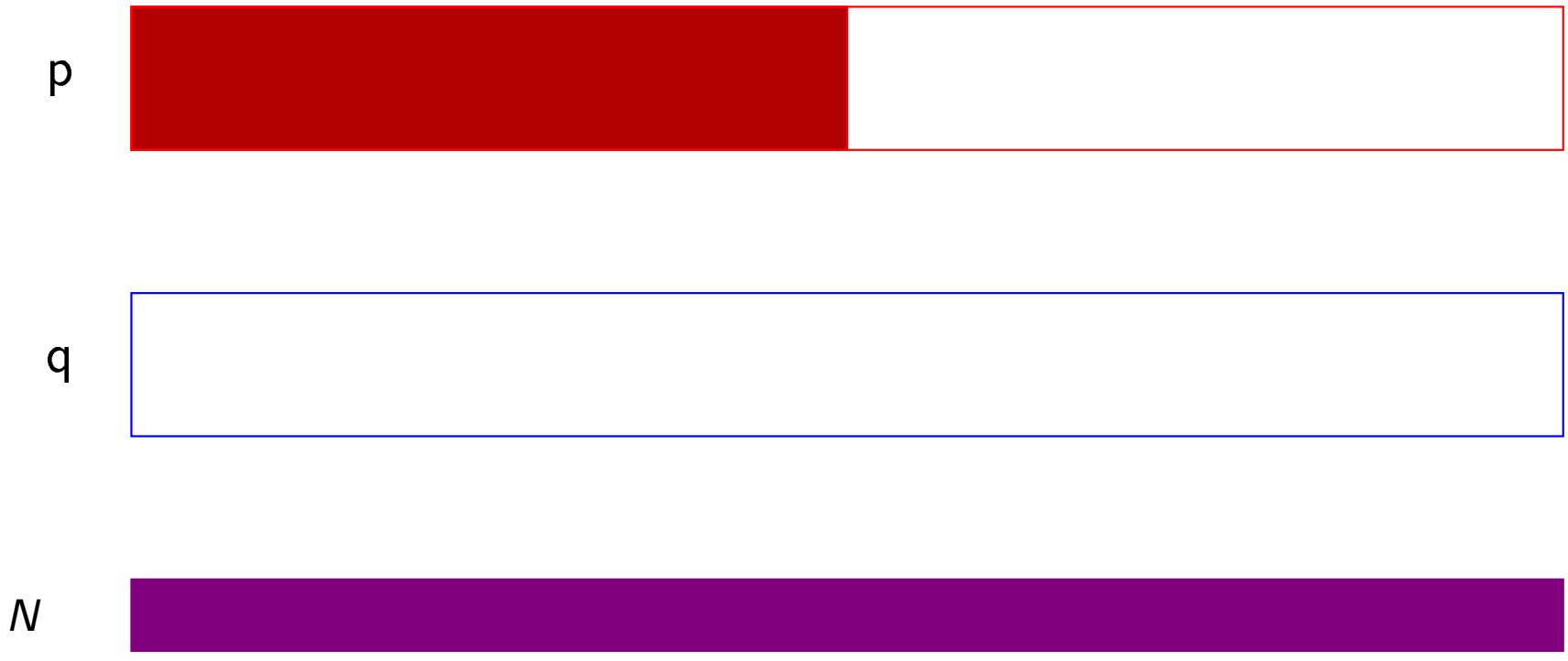
It is not possible to solve for $r > N^{1/d}$ with any method that constructs auxiliary polynomial $Q(x)$.

Countermeasures for real-world RSA

- Must use padding scheme with cryptographically secure randomized padding for RSA.
  - PKCS#1v1.5 widely used in practice, not CCA-secure.
  - OAEP is CCA-secure but not widely used.

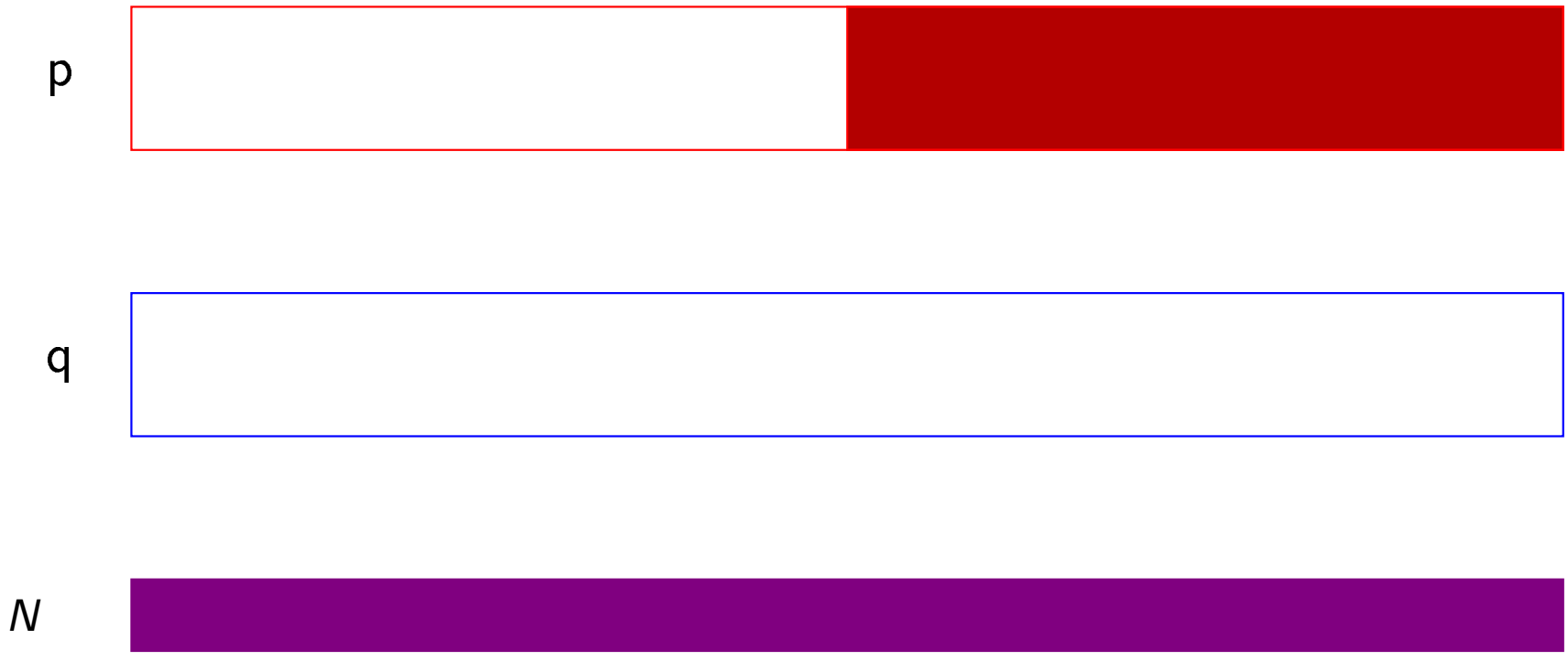
- Current recommendation: Use RSA exponent $e \geq 65537$.

Factoring with Partial Information



Polynomial time. (Lattice basis reduction.) [Coppersmith 96]

Factoring with Partial Information



Polynomial time. (Lattice basis reduction.) [Coppersmith 1996]

Theorem (Coppersmith 1996)

Let $N = pq$ with $p, q \approx \sqrt{N}$. Given half the bits (most or least significant) of p, we can factor N in polynomial time.

```
p = random_prime(2^512); q = random_prime(2^512)
N = p*q

a = p - (p % 2^86)
```

```
p = random_prime(2^512); q = random_prime(2^512)
N = p*q

a = p - (p % 2^86)

sage: hex(a)
'a9759e8c9fba8c0ec3e637d1e26e7b88befeb03ac199d1190
76e3294d16ffcaef629e2937a03592895b29b0ac708e79830
4330240bc00000000000000000000'
```

Key recovery from partial information.

```
p = random_prime(2^512); q = random_prime(2^512)
N = p*q

a = p - (p % 2^86)


X = 2^86
M = matrix([[X^2, X*a, 0], [0, X, a], [0, 0, N]])
B = M.LLL()
```

```
p = random_prime(2^512); q = random_prime(2^512)
N = p*q

a = p - (p % 2^86)


X = 2^86
M = matrix([[X^2, X*a, 0], [0, X, a], [0, 0, N]])
B = M.LLL()

Q = B[0][0]*x^2/X^2+B[0][1]*x/X+B[0][2]

sage: a+Q.roots(ring=ZZ)[0][0] == p
True
```

Partial key recovery and finding solutions modulo divisors

Assumption: Attacker doesn't know factorization of N.

Theorem (Howgrave-Graham)

Given degree d polynomial f, integer N, we can find roots r modulo divisors B of N satisfying

$$f(r) \equiv 0 \bmod B$$

for $|B| > N^{\beta}$, when $|r| < N^{\beta^2/d}$ in time polynomial in $\log N$ and d.

For RSA partial key recovery, we have

$$f(x) = a + x \quad d = 1 \quad \beta = 1/2$$

and we want to find a solution vanishing modulo $p \approx N^{1/2}$ for some $p \mid N$.

Efficient to solve for $|r| \leq N^{1/4}$.

Partial key recovery example

Input: $f(x) = a + x, N$
Output: $r < R \quad$ s.t. $\quad f(r) \equiv 0 \bmod p, \quad p|N, \quad p \geq N^{1/2}$

1. We chose the polynomial basis $x(x + a), (x + a), N$.

Partial key recovery example

Input: $f(x) = a + x, N$

Output: $r < R \quad \text{s.t.} \quad f(r) \equiv 0 \bmod p, \quad p|N, \quad p \geq N^{1/2}$

1. We chose the polynomial basis $x(x + a), (x + a), N$.
2. This corresponds to a lattice basis

$$\begin{bmatrix} R^2 & Ra & 0 \\ 0 & R & a \\ & & N \end{bmatrix}$$

$\dim L = 3$

$\det L = R^3 N$

Partial key recovery example

Input: $f(x) = a + x, N$
Output: $r < R \quad \text{s.t.} \quad f(r) \equiv 0 \bmod p, \quad p|N, \quad p \geq N^{1/2}$

1. We chose the polynomial basis $x(x+a), (x+a), N$.
2. This corresponds to a lattice basis

$$\begin{bmatrix} R^2 & Ra & 0 \\ 0 & R & a \\ & & N \end{bmatrix}$$

$$\dim L = 3$$
$$\det L = R^3 N$$

3. LLL will find us a vector of size about $|v| \approx \det L^{1/\dim L}$.

Partial key recovery example

Input: $f(x) = a + x, N$
Output: $r < R \quad \text{s.t.} \quad f(r) \equiv 0 \bmod p, \quad p|N, \quad p \geq N^{1/2}$

1. We chose the polynomial basis $x(x + a), (x + a), N$.
2. This corresponds to a lattice basis

$$\begin{bmatrix} R^2 & Ra & 0 \\ 0 & R & a \\ & & N \end{bmatrix} \qquad \begin{aligned} \dim L &= 3 \\ \det L &= R^3 N \end{aligned}$$

3. LLL will find us a vector of size about $|v| \approx \det L^{1/\dim L}$.
4. The algorithm will find the root when we have

$$\begin{aligned} |Q(r)| \leq |v| \approx \det L^{1/\dim L} &< p \\ (R^3 N)^{1/3} &< N^{1/2} \\ R &< N^{1/6} \end{aligned}$$

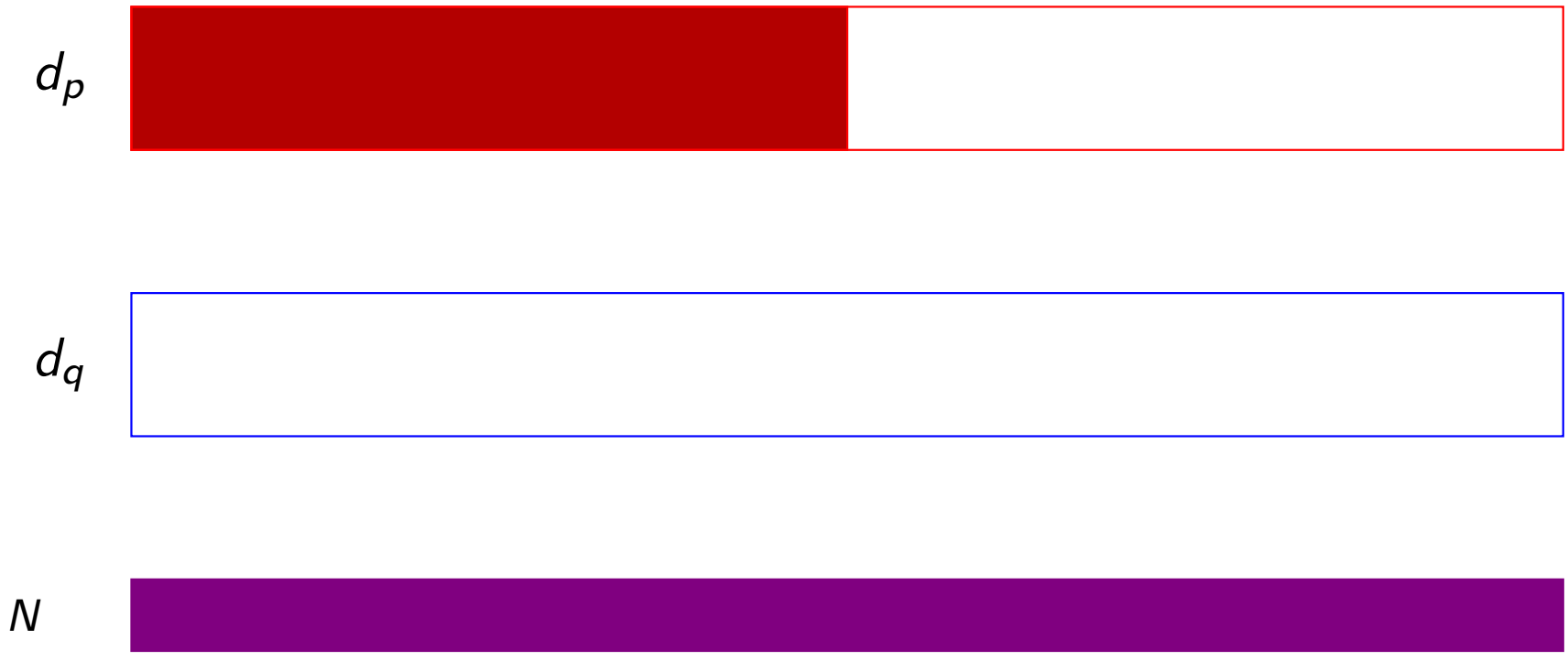
We had $\lg r = 86$ and $\lg p = 512$.

Partial key recovery and related attacks

RSA particularly susceptible to partial key recovery attacks.

- Can factor given 1/2 bits of p. [Coppersmith 96]
- Can factor given 1/4 bits of d. [Boneh Durfee Frankel 98]
- Can factor given 1/2 bits of $d \bmod (p-1)$. [Blömer May 03]

Factoring with Partial Information



Polynomial time. (Lattice basis reduction.) [Blömer May 03]

Key recovery from partial information on CRT-RSA

Let $d_p = d \pmod{p} - 1$.

Assume we know some a such that $d_p = a + r$ and r small.

$$\text{RSA equation:} \qquad ed_p = 1 + k_p(p-1)$$

$$\text{Rearrange:} \qquad (ed_p - 1 + k_p) = k_p p$$

Then we would like to solve for a small solution r to:

$$x + a - e^{-1}(1 + k_p) \equiv 0 \bmod p$$

For e small, we can brute force over k_p, and we know $p|N$.

We can apply Coppersmith/Howgrave-Graham technique as before.

```
p = random_prime(2^512); q = random_prime(2^512)
N = p*q

d = random_prime(2^254)
e = inverse_mod(d,(p-1)*(q-1))
```

d is relatively small. (But not that small.)

```
p = random_prime(2^512); q = random_prime(2^512)
N = p*q

d = random_prime(2^254)
e = inverse_mod(d,(p-1)*(q-1))



X = 2^764; Y = 2^254
M = matrix([[X, e*Y, -1], [0, Y*(N+1), 0], [0, 0, N+1]])

B = M.LLL()
```

```
p = random_prime(2^512); q = random_prime(2^512)
N = p*q

d = random_prime(2^254)
e = inverse_mod(d,(p-1)*(q-1))



X = 2^764; Y = 2^254
M = matrix([[X, e*Y, -1], [0, Y*(N+1), 0], [0, 0, N+1]])

B = M.LLL()

sage: abs(B[0][0]/X) == d
True
```

Small RSA private exponent with lattices

Theorem (Wiener)

We can efficiently compute d when $d < N^{1/4}$.

The *RSA equation* is

$$ed \equiv 1 \bmod (p-1)(q-1)$$
$$ed = 1 + k(N - (p+q) + 1)$$

Small RSA private exponent with lattices

Theorem (Wiener)

We can efficiently compute d when $d < N^{1/4}$.

The *RSA equation* is

$$ed \equiv 1 \bmod (p-1)(q-1)$$
$$ed = 1 + k(N - (p+q) + 1)$$

Let $s = p + q$.

We would like to solve

$$ed = 1 - ks + k(N+1)$$

for d, k, s unknown.

We know $k \leq d$ and $s \approx \sqrt{N}$.

Small RSA private exponent with lattices

We would like to solve

$$ed = 1 - ks + k(N + 1)$$

for d, k, s unknown.

Can write as

$$ks + ed - 1 \equiv 0 \bmod (N + 1)$$

We would like to find small solutions $x = ks, y = d$ for

$$f(x, y) = x + ey - 1 \equiv 0 \bmod (N + 1).$$

Small RSA private exponent with lattices

Would like to solve equation

$$f(x, y) = x + ey - 1 \equiv 0 \bmod (N + 1)$$

for solution $x = ks, y = d$. Bound $|d| < X$, $|ks| < Y$.

Create lattice basis

$$\begin{bmatrix} X & eY & -1 \\ & Y(N+1) & \\ & & (N+1) \end{bmatrix}$$

$\dim L = 3$
$\det L = XY(N+1)^2$

Corresponds to $x + ey - 1$, $y(N + 1)$, $(N + 1)$.

Small RSA private exponent with lattices

We will find a coefficient vector for a polynomial $Q(x, y)$ satisfying $Q(d, ks) = 0$ over $\mathbb{Z}$ when

$$|Q(d, ks)| \leq \dim L^{1/\det L} < N + 1$$
$$(XY(N+1)^2)^{1/3} < N + 1$$
$$XY < N + 1$$

We want to find solutions $d < X$, $ks < Y$, and we know

$$k \leq d \qquad s \approx \sqrt{N}$$

So when $d < X = N^{1/4}$, we can set $Y \approx N^{3/4}$ and $X \approx N^{1/4}$ and guarantee

$$Q(d, ks) = 0.$$

Small RSA private exponent with lattices

We will find a coefficient vector for a polynomial $Q(x, y)$ satisfying $Q(d, ks) = 0$ over $\mathbb{Z}$ when

$$|Q(d, ks)| \leq \dim L^{1/\det L} < N + 1$$
$$(XY(N+1)^2)^{1/3} < N + 1$$
$$XY < N + 1$$

We want to find solutions $d < X$, $ks < Y$, and we know

$$k \leq d \qquad s \approx \sqrt{N}$$

So when $d < X = N^{1/4}$, we can set $Y \approx N^{3/4}$ and $X \approx N^{1/4}$ and guarantee

$$Q(d, ks) = 0.$$

Lattice is actually finding equation

$$dx + (ks - 1)y - d = 0$$

Theorem (Boneh Durfee)

We can efficiently compute d when $d < N^{0.292}$.

Boneh and Durfee use Coppersmith's method to find small solutions $x = k$, $y = (p + q)$ to

$$xy - (N + 1)x - 1 \equiv 0 \bmod e$$

Improvements: Use higher multiplicities and degree, examine sublattice.

ECDSA signature scheme

Public Parameters

- An elliptic curve E
- A base point G of order n on E.

Private Key

- An integer d mod n.

Public Key

- $Q = dG$ in uncompressed (x, y) or compressed $(x, 1 \text{ bit of } y)$ format.

Sign

1. Input message hash h.
2. Choose integer k mod n.
3. Compute point $(r, y_r) = kG$.
4. Output $(r, s = k^{-1}(h + dr) \bmod n)$.

Partial key recovery for (EC)DSA:

An attacker learns some information about the signature nonce k.
Can they efficiently recover the secret key d?

ECDSA key recovery from nonce k: You just did this.

k 

Sign

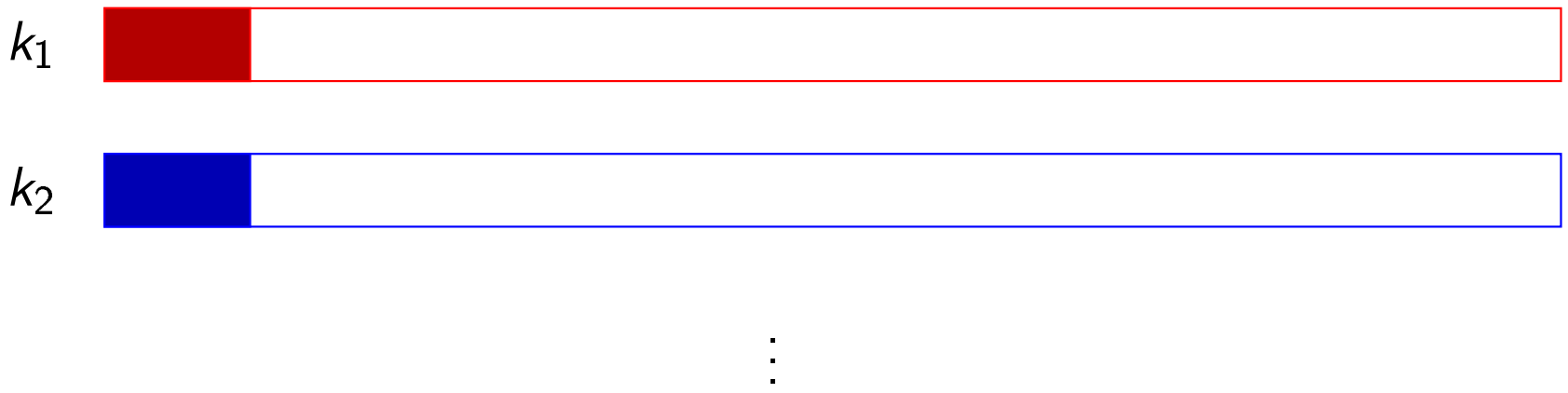
1. Input message hash h.
2. Choose integer k mod n.
3. Compute point $(r, y_r) = kG$.
4. Output $(r, s = k^{-1}(h + dr) \bmod n)$.

Fact

If an attacker learns k for a signature, the long-term secret key d is revealed.

$$d = (sk - h)r^{-1} \bmod n$$

ECDSA key recovery from partial information about nonces



Polynomial time, using lattices. [Howgrave-Graham Smart 2001], [Nguyen Shparlinski 2003]

ECDSA key recovery from partial information about nonces

Secret key d can be computed from MSBs of nonces.

Input signatures $(r_1, s_1), \ldots, (r_m, s_m)$ on messages $h_1, \ldots, h_m$.

Then we have a system of equations in unknowns $k_1, \ldots, k_m, d$:

$$
\begin{aligned}
k_1 - s_1^{-1} r_1 d - s_1^{-1} h_1 &\equiv 0 \bmod n \\
k_2 - s_2^{-1} r_2 d - s_2^{-1} h_2 &\equiv 0 \bmod n \\
&\vdots \\
k_m - s_m^{-1} r_m d - s_m^{-1} h_m &\equiv 0 \bmod n
\end{aligned}
$$

ECDSA key recovery from partial information about nonces

Secret key d can be computed from MSBs of nonces.

Input signatures $(r_1, s_1), \ldots, (r_m, s_m)$ on messages $h_1, \ldots, h_m$.

Assume we have learned MSBs of k_i so that $k_i = a_i + b_i$ with $b_i < B$.

Then we have a system of equations in unknowns $b_1, \ldots, b_m, d$:

$$
\begin{aligned}
b_1 - s_1^{-1} r_1 d + a_1 - s_1^{-1} h_1 &\equiv 0 \bmod n \\
b_2 - s_2^{-1} r_2 d + a_2 - s_2^{-1} h_2 &\equiv 0 \bmod n \\
&\vdots \\
b_m - s_m^{-1} r_m d + a_m - s_m^{-1} h_m &\equiv 0 \bmod n
\end{aligned}
$$

Formulating ECDSA as a hidden number problem

[Howgrave-Graham Smart 2001], [Nguyen Shparlinski 2003]

We have a system of equations in unknowns $b_1, \ldots, b_m, d$:

$$\begin{aligned} b_1 - t_1 d - u_1 &\equiv 0 \bmod n \\ b_2 - t_2 d - u_2 &\equiv 0 \bmod n \\ &\vdots \\ b_m - t_m d - u_m &\equiv 0 \bmod n \end{aligned}$$

We assume the b_i are small.

This is an instance of the *hidden number problem* [Boneh Venkatesan 96].

Solving the hidden number problem with lattices

Input:

$$b_1 - t_1 d - u_1 \equiv 0 \bmod n$$
$$\vdots$$
$$b_m - t_m d - u_m \equiv 0 \bmod n$$

in unknowns $b_1, \ldots, b_m, d$, where $|b_i| < B$.

Construct the lattice

$$M = \begin{bmatrix} n & & & & & \\ & n & & & & \\ & & \ddots & & & \\ & & & n & & \\ t_1 & t_2 & \ldots & t_m & B/n & \\ u_1 & u_2 & \ldots & u_m & & B \end{bmatrix}$$

$v_k = (b_1, b_2, \ldots, b_m, Bd/n, B)$ is a short vector in this lattice.

Solving the hidden number problem with lattices

Construct the lattice

$$M = \begin{bmatrix} n & & & & & \\ & n & & & & \\ & & \ddots & & & \\ & & & n & & \\ t_1 & t_2 & \dots & t_m & B/n & \\ u_1 & u_2 & \dots & u_m & & B \end{bmatrix}$$

Want vector
$v_k = (b_1, b_2, \dots, b_m, Bd/n, B)$

We have:

- $\dim L = m + 2$ $\qquad\qquad$ $\det L = B^2 n^{m-1}$
- Ignoring approximation factors, LLL or BKZ will find a vector

$$|v| \leq (\det L)^{1/\dim L}$$

- We are searching for a vector with length $|v_k| \leq \sqrt{m+2}B$.
- Thus we expect to find v_k when

$$\log B \leq \lfloor \log n(m-1)/m - (\log m)/2 \rfloor$$

Solving the hidden number problem with lattices

We expect to find v_k when

$$\log B \leq \lfloor \log n(m-1)/m - (\log m)/2 \rfloor$$

- 160-bit n: Learn key from 2 bits of information per signature by reducing 100-dimensional lattice [LN 13]
- 256 bit n: Learn key from 3 bits of information per signature by reducing 100-dimensional lattice

Summary

Lattices are a powerful tool especially in side-channel attack scenarios where an attacker can observe partial information about secrets like keys or nonces.