Cryptanalysis via Lattice Techniques

Alexander May

Horst Görtz Institute for IT-Security
Faculty of Mathematics
Ruhr-University Bochum

crypt@b-it 2010, Aug 2010, Bonn

Motivation

Ultimate goal Find roots of multivariate polynomials

Given: polynomial $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$

Find : solutions $(x_1^{(0)}, \ldots, x_n^{(0)}) \in \mathbb{Z}^n$ with $f(x_1^{(0)}, \ldots, x_n^{(0)}) = 0 \bmod N$

Examples:

- Factorization problem $N = pq$:
$$f(x, y) = N - xy \text{ with } (x^{(0)}, y^{(0)}) = (p, q)$$
- RSA equation $ed = 1 \bmod \phi(N) \Leftrightarrow ed = 1 + k(N - (p + q - 1))$:
$$f(x, y, z) = ex - 1 - y(N - z),\ (x^{(0)}, y^{(0)}, z^{(0)}) = (d, k, p + q - 1)$$

Goal 1 Find small modular roots of linear polynomials

Given: linear $f(x_1, \ldots, x_n) = a_1x_1 + a_2x_2 + \ldots + a_nx_n$, modulus N

Find : small solutions $(x_1^{(0)}, \ldots, x_n^{(0)})$ with $f(x_1^{(0)}, \ldots, x_n^{(0)}) = 0 \bmod N$

First definition of a lattice

Definition 1 Lattice

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

Properties:

- Closed: $\mathbf{u}, \mathbf{v} \in L \Rightarrow \mathbf{u} + \mathbf{v} \in L$
- Neutral element: $\mathbf{0} = O^n \in L$
- Inverse element: $\mathbf{u} \in L \Rightarrow -\mathbf{u} \in L$
- Discrete: no accumulation point

Examples:

- $\mathbb{Z} \subset \mathbb{R}$ is a lattice.
- $k\mathbb{Z} \subset \mathbb{R}$ is a lattice.
- $\mathbb{Z}^d \subset \mathbb{R}^n$, $d \leq n$ is a lattice.
- $(k\mathbb{Z})^d \subset \mathbb{R}^n$, $d \leq n$ is a lattice.

Representation problem: Lattices have an infinite number of points.

Second definition of a lattice

Definition 2 Lattice

Let $\mathbf{b_1}, \mathbf{b_2}, \dots, \mathbf{b_d} \in \mathbb{R}^n$ be linearly independent. Then

$$L = \left\{ \mathbf{v} \in \mathbb{R} \mid \mathbf{v} = \sum_{i=1}^{d} a_i \mathbf{b_i}, a_i \in \mathbb{Z} \right\} \text{ is a lattice.}$$

Exercise: Show that both definitions are equivalent.

Notation for lattices:

- **Basis** $B = \begin{pmatrix} \mathbf{b_1} \\ \vdots \\ \mathbf{b_d} \end{pmatrix} \in \mathbb{R}^{d \times n}$ with **rank** d and **dimension** n.
- Lattice has **full rank** if $d = n$.

Non-uniqueness of bases

Definition Unimodular transformations

Let B be a basis. Unimodular transformations of B consist of

1. permutation of basis vectors,
2. addition of a multiple of a basis vector to another basis vector.

Exercise: Unimodular transformations leave lattice unchanged.

Exercise: Unimodular transformations are multiplications $T \cdot B$ with

$$T \in \mathbb{Z}^{d \times d}, \det(T) = \pm 1.$$

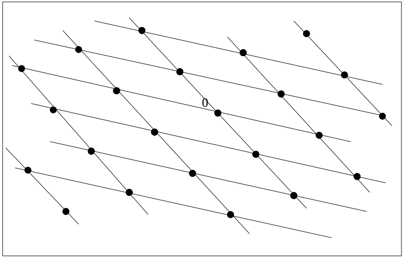
Theorem

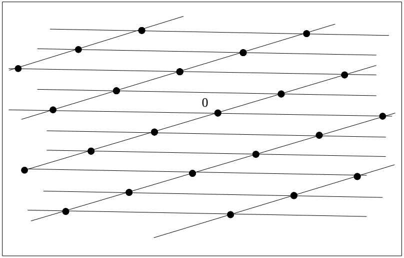
Let L be a lattice. Then L has infinitely many bases.

Proof: There exist infinitely many unimodular transformations.

Good bases: Short and pairwise almost orthogonal basis vectors.

Two different bases of the same lattice





The lattice determinant

Definition Lattice determinant

Let L be a full rank lattice with basis B. The lattice determinant $\det(L)$ is defined as $\det(L) := |\det(B)|$.

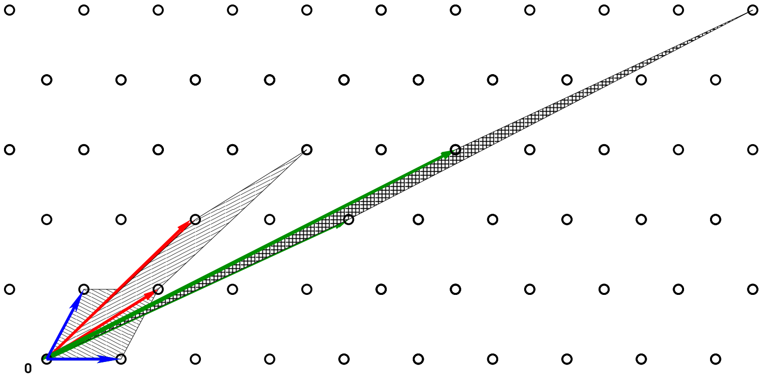
Property:

- For unimodular T: $|\det(TB)| = |\det(T) \cdot \det(B)| = |\det(B)|$.
- That means $\det(L)$ is a *lattice invariant*.

Geometric interpretation: Fundamental region $P(B)$

- Let $P(B) = \left\{ \mathbf{v} \in \mathbb{R}^n \mid \mathbf{v} = \sum_{i=1}^{d} x_i \mathbf{b_i}, x_i \in \mathbb{R}, 0 \leq x_i \leq 1 \right\}$.
- Then $\det(L)$ is the volume of the fundamental region $P(B)$.

The lattice determinant is an invariant.



Back to linear equations

Lemma

The set of integer solutions of $a_1x_1 + \ldots + a_nx_n = 0 \bmod N$ forms a lattice of rank n.

Proof: Check via Definition 1 of a lattice.

- $\mathbf{x} = (x_1, \ldots, x_n) = 0^n$ is a solution.
- Let $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^n$ be solutions. Then $\mathbf{u} - \mathbf{v}$ is a solution.
- Let $\mathbf{e_i}$ be the unit vectors. Since $N\mathbf{e_i}, i = 1 \ldots n$, are n linearly independent solutions, the lattice rank is at least n.
- Since the solutions are in $\mathbb{Z}^n$ the rank is at most n.

Exercise 1: Find a basis for the lattice.

Exercise 2: Let $A \in \mathbb{Z}^{m \times n}$ have rank m. Then $\{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{x}A = 0\}$ forms a lattice of rank $n - m$.

Successive minima

Definition Successive minima

Let L be a rank d lattice. For $i \leq d$ we denote by λ_i the minimal radius of a ball around $\mathbf{0}$ that contains i linearly independent vectors.

Theorem

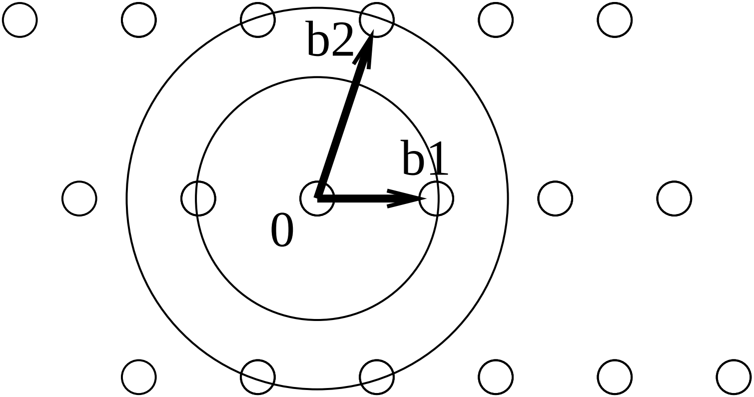
Let L be a rank d lattice with $d \geq 5$. Then d linearly independent vectors do not necessarily form a basis of L.

Proof: Let L be spanned by the basis

$$B = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

- L contains $2\mathbf{e_i}$ for $i = 1, \ldots, 5$. Therefore, $\lambda_1 = \ldots = \lambda_5 = 2$.
- But $2I_5$ is not a basis of L, since it does not contain (1,1,1,1,1).

Successive minima $\lambda_1 = \|b_1\|, \lambda_2 = \|b_2\|$



Minkowski's Theorem

Theorem of Minkowski

Let L be a rank d lattice. Then $\lambda_1 \leq \sqrt{d} \cdot \det(L)^{\frac{1}{d}}$.

Heuristic 1

Let L be a rank d lattice. Let $\mathbf{v} \in L$ with $\|\mathbf{v}\| \ll \sqrt{d} \det(L)^{\frac{1}{d}}$. Then $\mathbf{v}$ is a shortest vector in L.

Algorithmic problems: SVP and CVP

Problem Shortest Vector Problem (SVP)

Given: $B \in \mathbb{Q}^{d \times n}$ for L

Find: $\mathbf{v} \in L \setminus \mathbf{0}$ with $\|\mathbf{v}\| = \lambda_1$ (or $\|\mathbf{v}\| \leq \gamma\lambda_1$ for approx factor γ)

Problem Closest Vector Problem (SVP)

Given: $B \in \mathbb{Q}^{d \times n}$ for L, target $\mathbf{t} \in \mathbb{Q}^n$

Find: $\mathbf{v} \in L$ with $\|\mathbf{v} - \mathbf{t}\| = \min_{u \in L} \|\mathbf{u} - \mathbf{t}\|$

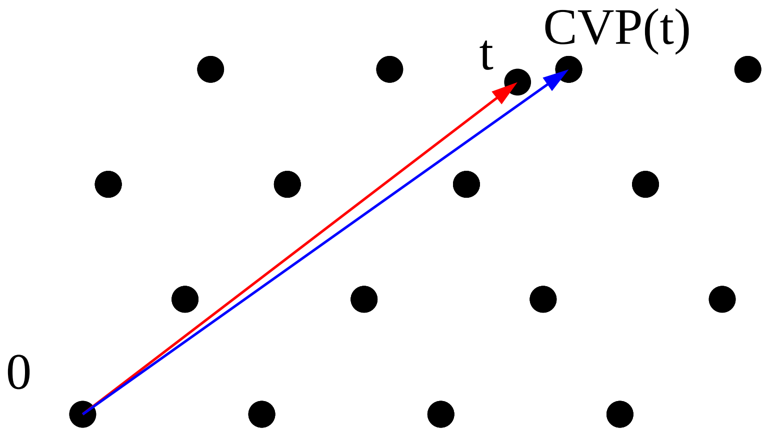
(or $\|\mathbf{v} - \mathbf{t}\| \leq \gamma \cdot \min_{u \in L} \|\mathbf{u} - \mathbf{t}\|$ for approx factor γ)

Theorem

1. CVP is NP-hard (van Emde Boas 1981)
2. SVP is NP-hard (Ajtai 1996)

Unlikely: Algorithm with run time poly in $d, n, \log b_{\max} = \log(max_{i,j} b_{i,j})$.

Closest Vector Problem (CVP)



Lattice reduction with fixed rank d

Algorithm Gauß (rank 2)

INPUT: basis $\mathbf{b_1}, \mathbf{b_2} \in \mathbb{Q}^n$ with $\|\mathbf{b_1}\| \geq \|\mathbf{b_2}\|$

1. Find $k \in \mathbb{Z}$ that minimizes $\|\mathbf{b_1} - k\mathbf{b_2}\|$. Set $\mathbf{b_1} \leftarrow \mathbf{b_1} - k\mathbf{b_2}$.
2. If $k \neq 0$, swap $\mathbf{b_1}$ and $\mathbf{b_2}$.

OUTPUT: basis $\mathbf{b_1}, \mathbf{b_2}$ with $\|\mathbf{b_1}\| = \lambda_1$, $\|\mathbf{b_2}\| = \lambda_2$

Running time: $\mathcal{O}(n \log^2 b_{\max})$

Example:

- On input $\mathbf{b_1} = (11, 6), \mathbf{b_2} = (8, 4)$, the Gauß algorithm outputs

$$\mathbf{b_1} = (2, 0), \mathbf{b_2} = (1, 2).$$

Theorem

Let $B \in \mathbb{Q}^{d \times n}$ be a lattice basis. Then SVP and CVP can be solved in time polynomial in $(n, \log b_{\max})$.

Approximative SVPs in arbitrary dimension

Theorem LLL algorithm (Lenstra, Lenstra, Lovász 1982)

Let L be a lattice with basis $\mathbf{v_1}, \ldots, \mathbf{v_d} \in \mathbb{Q}^n$. Then the LLL algorithm computes a basis $\mathbf{b_1}, \ldots, \mathbf{b_d}$ with

1. $\|\mathbf{b_1}\| \leq c^d \cdot \det(L)^{\frac{1}{d}}$
2. $\|\mathbf{b_i}\| \leq c^{2d} \cdot \lambda_i(L)$

where $c = \left(\frac{4}{3}\right)^{\frac{1}{4}} \approx 1.075$ in time $\mathcal{O}(d^5 n \log^3 b_{\max})$.

Solving linear equations

Goal 1: Find small modular roots of linear polynomials

Given: $a_1, \ldots, a_n \in \mathbb{Z}_n, N \in \mathbb{N}$ with $\gcd(a_i, N) = 1$ for some i and $a_1x_1 + \ldots a_nx_n = 0 \bmod N$ for unknown $(x_1, \ldots, x_n) \in \mathbb{Z}^n$, upper bounds $X_i \in \mathbb{Z}$ such that $|x_i| \leq X_i$ and $\prod_{i=1}^{n} X_i \leq N$.

Find : small solution $\mathbf{x} = (x_1, \ldots, x_n)$ as solution of SVP

A first approach

- Wlog $\gcd(a_n, N) = 1$. Set $b_i := -a_i \cdot a_n^{-1}$. We obtain
$$b_1 x_1 + \ldots + b_{n-1} x_{n-1} = x_n \bmod N.$$
- Create lattice L spanned by the basis
$$B = \begin{pmatrix} 1 & & & & b_1 \\ & 1 & & & b_2 \\ & & \ddots & & \vdots \\ & & & 1 & b_{n-1} \\ & & & & N \end{pmatrix}.$$
- We have $\mathrm{rank}(L) = n$, $\det(L) = \det(B) = N$.
- Let $b_1 x_1 + \ldots + b_{n-1} x_{n-1} = x_n - yN$ for some $y \in \mathbb{Z}$.
- Then $(x_1, \ldots, x_{n-1}, y) \cdot B = (x_1, \ldots, x_{n-1}, \underbrace{\sum_{i=1}^{n-1} b_i x_i + yN}_{x_n}) = \mathbf{x}$.
- Thus $\mathbf{x} \in L$ with $\|\mathbf{x}\| \leq \sqrt{n} \cdot \max_i\{x_i\}$.
- Minkowski bound: $\lambda_1 \leq \sqrt{n} \cdot N^{\frac{1}{n}}$.
- Iff $x_1 \approx \ldots \approx x_n \approx N^{\frac{1}{n}}$, then $\mathbf{x}$ is a short vector (Heuristic 1).

A second approach

- Wlog $\prod_{i=1}^{n} X_i = N$. Multiply ith column vector of B with $Y_i := \frac{N}{X_i}$.

$$B' = \begin{pmatrix} Y_1 & & & & Y_n b_1 \\ & Y_2 & & & Y_n b_2 \\ & & \ddots & & \vdots \\ & & & Y_{n-1} & Y_n b_{n-1} \\ & & & & Y_n N \end{pmatrix}.$$

- We obtain $rank(L') = n$ and

$$\det(L') = N \cdot \prod_{i=1}^{n} Y_i = N \cdot \prod_{i=1}^{n} \frac{N}{X_i} = N^{n+1} \prod_{i=1}^{n} \frac{1}{X_i} = N^n.$$

- Now $(x_1, \ldots, x_{n-1}, y) \cdot B' = (x_1 Y_1, \ldots, x_{n-1} Y_{n-1}, x_n Y_n) = \mathbf{x}'$.
- We have $|x_i| Y_i \le \frac{|x_i|}{X_i} \cdot N < N$ and thus $\mathbf{x}' < \sqrt{n} \cdot N$.
- Minkowski bound: $\lambda_1(L') \le \sqrt{n} \det(L')^{\frac{1}{n}} = \sqrt{n} \cdot N$.
- Under Heuristic 1, we can expect to find $\mathbf{x}'$ as the solution of an SVP.
- From $\mathbf{x}'$ we can easily recover the desired solution vector $\mathbf{x}$.

Solving inhomogenous or non-modular equations

Problem Inhomogenous equation

Find solution of $a_1x_1 + \ldots + a_nx_n = b \bmod N$.

Approach via CVP instance

- Define rank $n+1$ lattice with vectors $(x_1, \ldots, x_n, \sum_{i=1}^n a_ix_i - yN)$.
- Define CVP target vector as $(0, \ldots, 0, b)$.

Exercise: Find a variation that uses an SVP instance as before.

Problem Equation over the integers

Find solution of $a_1x_1 + \ldots + a_nx_n = b$.

Approach:

- Reduce modulo largest of a_i or b. We are back to modular case.

Wiener attack

Theorem Wiener (1990)

Let $N = pq$ with p, q of equal bit-size. Let $ed = 1 \bmod \phi(N)$ with $d \leq \frac{1}{3}N^{\frac{1}{4}}$. Under Heuristic 1, N can be factored in time $\mathcal{O}(\log^2 N)$.

Proof:

- Write $ed = 1 \bmod \phi(N)$ as $ed = 1 + k(N - (p + q - 1))$, $k \in \mathbb{N}$, with
$$k = \frac{ed-1}{\phi(N)} \leq \frac{e}{\phi(N)} \cdot d < d.$$
- Write the RSA equation as $ed + k(p + q - 1) - 1 = kN$.
- Linearization: $ex_1 + x_2 = 0 \bmod N$ with $(x_1, x_2) = (d, k(p + q - 1))$.
- We can define $X_1 = \frac{1}{3}N^{\frac{1}{4}}$.
- In order to define X_2 we observe that wlog $p < \sqrt{N} < q$,
$$q < 2p < 2\sqrt{N} \text{ and therefore } p + q < 3\sqrt{N}.$$
- Define an upper bound of $k(p + q - 1) < d(p + q - 1) < N^{\frac{3}{4}} =: X_2$.

Wiener attack

Proof (continued):

- The requirements of our lattice method are fulfilled, since

  $X_1X_2 < N$ and the coefficient of x_2 is co-prime to N.
- Under Heuristic 1, we find (x_1, x_2) as solution of an SVP instance.
- We use a lattice with rank 2. (Exercise: Construct a basis.)
- Running time of the Gauß algorithm is $\mathcal{O}(\log^2 b_{\max}) = \mathcal{O}(\log^2 N)$.
- From (x_1, x_2) be obtain d, $k = \frac{ex_1+x_2}{N}$ and $\phi(N) = \frac{ex_1-1}{k}$.
- From $\phi(N)$ and N we can easily derive p, q (Exercise).

Attacking GnuPG ElGamal signatures

El Gamal signature

1. **Params:** public: prime p, α generator of $\mathbb{Z}_p^*$, $\beta = \alpha^a \bmod p$
   private: $a \in \mathbb{Z}_{p-1}$
2. **Sign:** $\sigma(m) = (\gamma, \delta) = (\alpha^r \bmod p, r^{-1}(m - a\gamma) \bmod p - 1)$
   In GnuPG: $a, r < p^{\frac{3}{8}}$ for efficiency reasons

Linearization attack (Nguyen 2004):

- Write $\delta = r^{-1}(m - a\gamma)$ as
$$\delta r + \gamma a = m \bmod p - 1.$$
- We obtain a linear modular equation in the unknowns r and a.
- The product $ra \leq p^{\frac{3}{4}} \ll p - 1$ satisfies our size restriction.
- If $\gcd(\delta, p - 1)$ or $\gcd(\gamma, p - 1) = 1$, we can apply our method.
- Under Heuristic 1, we find (r, a) by solving SVP in a rank 3 lattice.

Pseudo Random Number Generators (PRNGs)

Algorithm Linear Congruential

1. **Params:** public: $N \in \mathbb{N}$, secret: $a, b, x_0 \in \mathbb{Z}_N$
2. **Alg:** Iterate $x_{i+1} = ax_i + b \bmod N$, $i = 0, 1, \ldots$.
   Output a fraction of the most significant bits of x_{i+1}.

Properties:

- Easy: x_1, x_2, x_3 allow for computing the whole sequence.
- Broken for every fixed fraction of output bits via lattice method. (Hastad, Shamir 1985)

Algorithm Pollard Generator

1. **Params:** public: prime $p \in \mathbb{N}$, secret: $b, x_0 \in \mathbb{Z}_N$
2. **Alg:** Iterate $x_{i+1} = x_i^2 + b \bmod N$, $i = 0, 1, \ldots$.
   Output a fraction of the most significant bits of x_{i+1}

Question: When do we output too many bits?

Attacking the Pollard Generator

Theorem Blackburn, Gomez-Perez, Gutierrez, Shparlinski 2005

Let r_1, r_2, r_3 be the output of the Pollard generator with $|x_i - r_i| < \frac{1}{2}p^{\frac{1}{4}}$. Then the whole sequence can be computed efficiently.

Proof:

- We have $\left| \begin{array}{ccc} x_2 & = & x_1^2 + c \bmod p \\ x_3 & = & x_2^2 + c \bmod p \end{array} \right| \Rightarrow x_2 - x_3 = x_1^2 - x_2^2 \bmod p.$
- Let $x_i = r_i + y_i$ with $|y_i| \leq \frac{1}{2}p^{\frac{1}{4}}$. Our goal is to recover the y_i.

$$\begin{array}{rl} & r_2 + y_2 - r_3 - y_3 = (r_1 + y_1)^2 - (r_2 + y_2)^2 \bmod p \\ \Rightarrow & \underbrace{y_2^2 - y_1^2 + y_2 - y_3}_{z} + 2r_2y_2 - 2r_1y_1 = \underbrace{r_1^2 - r_2^2 + r_3 - r_2}_{c} \bmod p. \end{array}$$

- We obtain a linear, inhom equation in z, y_2, y_1. Coefficient of z is 1.
- Can apply SVP in a rank 4 lattice provided that $|zy_1y_2| \leq p$.
- The size restriction is satisfied, since we have $|y_1|, |y_2| \leq p^{\frac{1}{4}}$ and

$$|z| \leq |y_2^2| + |y_1^2| + |y_2| + |y_3| \leq \frac{1}{2}p^{\frac{1}{2}} + p^{\frac{1}{4}} \leq p^{\frac{1}{2}}.$$

- From y_1, y_2 we obtain x_1, x_2 which in turn yields $c = x_2 - x_1^2 \bmod p$.

Solving polynomial modular equations

Goal 2 Find small modular roots of polynomials

Given: integer N of unknown factorization, monic polynomial

$$f(x) = x^n + a_{n-1} + \ldots + a_1 x + a_0$$

Find: all small roots x_0 with $f(x_0) = 0 \bmod N$

Remark: Finding all root in $\mathbb{Z}_N$ is hard under the RSA assumption.

- Let $c = m^e \bmod N$ be an RSA ciphertext.
- The root $x_0 = m$ is the unique root of $f(x) = x^e - c \bmod N$.

Linearization:

- Linearize $f(x) = x_n + a_{n-1}x_{n-1} + \ldots + a_1 x_1 + a_0$ with $x_i := x^i$.
- Size restriction is $\prod_{i=1}^n x_i = \prod_{i=1}^n x^i \leq \prod_{i=1}^n X^i = X^{\frac{n(n+1)}{2}} \leq N$.
- This yields the bound $X \leq N^{\frac{2}{n(n+1)}}$.
- Requires to solve SVP in a rank $(n+1)$ lattice.

Coppersmith's method (1996)

Properties:

1. It suffices to compute a short vector via LLL instead of solving SVP. I.e., that the method stays poly time for non-constant n.
2. *Provably* (without heuristic) yields *all* sufficiently small roots.

Idea of Coppersmith's method:

Let $f(x) \in \mathbb{Z}[x]$.

Goal: Find all roots x_0 with $f(x_0) = 0 \bmod M$ and $|x_0| \leq X$. Maximize X.

1. Choose $m \in \mathbb{N}$. Define collection $f_1(x), \ldots, f_k(x)$ satisfying
$$f_i(x_0) = 0 \bmod M^m \text{ for } i = 1, \ldots, k.$$
Example: Choose $f_i(x) = x^i \cdot f(x)^m$.
2. Construct $g(x) = \sum_{i=1}^{k} a_i f_i(x)$ for $a_i \in \mathbb{Z}$ with
$$g(x_0) = 0 \text{ over } \mathbb{Z} \text{ for all } |x_0| \leq X.$$
Sufficient condition: $|g(x_0)| < M^m$.
3. Find root of $g(x)$ over $\mathbb{Z}$ with standard techniques.

Lemma of Hastad and Howgrave-Graham

Definition Norm of a polynomial

Let $g(x) = \sum_i a_i x^i \in \mathbb{Z}[x]$. Then the norm of g is $\|g\| = \sqrt{\sum_i a_i^2}$.

Lemma Howgrave-Graham

Let $g(x) \in \mathbb{Z}[x]$ with n monomials. Let $x_0 \in \mathbb{Z}$ with $|x_0| \leq X$. Further let

1. $g(x_0) = 0 \bmod M^m$,
2. $\|g(xX)\| < \frac{M^m}{\sqrt{n}}$.

Then $g(x_0) = 0$.

Proof:

$$\begin{aligned} |g(x_0)| &= \left|\sum_i a_i x_0^i\right| \leq \sum_i \left|a_i X^i \left(\frac{x_0}{X}\right)^i\right| \\ &\leq \sum_i \left|a_i X^i\right| \leq \sqrt{n} \cdot \|g(xX)\| < M^m. \end{aligned}$$

This implies

$$\left| \begin{array}{l} g(x_0) = k \cdot M^m \\ |g(x_0)| < M^m \end{array} \right| \Rightarrow g(x_0) = 0.$$

Theorem of Coppersmith

Theorem Coppersmith

Let $\epsilon > 0$. For sufficiently large $M \in \mathbb{N}$ the following holds. Let $f(x)$ be a monic polynomial of degree n. Then one can compute all roots x_0 with

$$f(x_0) = 0 \bmod M \text{ and } |x_0| \leq M^{\frac{1}{n}-\epsilon}$$

in time polynomial in $\log M, n$ and $\frac{1}{\epsilon}$.

Proof:

- Fix m. $\quad (m = \lceil \frac{1}{n\epsilon} \rceil)$
- Define collection
$$f_{i,j}(x) = x^j M^{m-i} f^i \text{ for } i = 0, \ldots, m-1, j = 0, \ldots, n-1.$$
- Let $f(x_0) = 0 \bmod M$. Then
$$f^i(x_0) = 0 \bmod M^i \text{ and } M^{m-i} f^i(x_0) = 0 \bmod M^m.$$
- This implies that $f_{i,j}(x_0) = 0 \bmod M^m$ and therefore
$$g(x_0) = \sum_{i,j} a_{i,j} f_{i,j}(x_0) = 0 \bmod M^m.$$

Theorem of Coppersmith

Proof: (continued)

- Order the polynomials $f_{i,j}(xX)$ in increasing order of their degree.

$$\begin{array}{l} f_{0,0}(xX), f_{0,1}(xX), \ldots, f_{0,n-1}(xX) \\ f_{1,0}(xX), f_{1,1}(xX), \ldots, f_{1,n-1}(xX) \\ \vdots \\ f_{m-1,0}(xX), f_{m,1}(xX), \ldots, f_{m-1,n-1}(xX) \end{array}$$

- Write the coefficient vectors of $f_{i,j}(xX)$ into a basis matrix

$$\begin{pmatrix} M^m & & & & & & & & \\ & \ddots & & & & & & & \\ & & M^m X^{n-1} & & & & & & \\ - & \ldots & - & M^{m-1}X^n & & & & & \\ & & \ddots & & \ddots & & & & \\ & & - & \ldots & - & MX^{(m-1)n} & & & \\ & & & \ddots & & \ddots & & & \\ & & & - & \ldots & - & MX^{mn-1} \end{pmatrix}$$

Theorem of Coppersmith

Proof: (continued)

- B spans a lattice L with $\mathrm{rank}(L) = mn$ and
$$\det(L) = M^{\frac{m(m+1)}{2}n} X^{\frac{(mn-1)mn}{2}} \approx M^{\frac{m^2 n}{2}} X^{\frac{m^2 n^2}{2}}.$$
- Every linear combination $\mathbf{v} = \mathbf{c} \cdot B$ defines a coefficient vector of some $g(xX)$ with no more than $\mathrm{rank}(L) = mn$ monomials.
- According to Howgrave-Graham's lemma we need
$$\|\mathbf{v}\| = \|g(xX)\| \leq \frac{M^m}{\sqrt{mn}}.$$
- The LLL algorithm computes a vector $\mathbf{v}$ with
$$\|\mathbf{v}\| \leq c^{\mathrm{rank}(L)} \cdot \det(L)^{\frac{1}{\mathrm{rank}(L)}}(L) \leq \frac{M^m}{\sqrt{mn}}.$$
- For sufficiently large M we can neglect $c^{\mathrm{rank}(L)}$ and $\sqrt{mn}$:
$$\begin{aligned} \det(L) \leq M^{m \cdot \mathrm{rank}(L)} \quad &\Leftrightarrow \quad M^{\frac{m^2 n}{2}} X^{\frac{m^2 n^2}{2}} \leq M^{m \cdot mn} \\ &\Leftrightarrow \quad X^{\frac{m^2 n^2}{2}} \leq M^{\frac{m^2 n}{2}} \qquad \Leftrightarrow X \leq M^{\frac{1}{n}}. \end{aligned}$$

Note: With some extra work approximations and the ϵ can be omitted.

Attack on stereotyped messages (Coppersmith 96)

Scenario:

- An attacker knows a stereotype part S of the message $m = S + x$.
- For example, $S =$"The codeword for today is".

Theorem

Let $m = S + x$ with known S. Then x can be computed from $c = m^e \bmod N$ in time polynomial in $(\log N, e)$ provided that $|x| \leq N^{\frac{1}{e}}$.

Proof:

- We want to find the unique root of the polynomial

$$f(x) = (S + x)^e - c \bmod N.$$

- Notice that $f(x)$ is a monic modular polynomial of degree $n = e$.
- Coppersmith's Theorem immediately yields the bound $|x| \leq N^{\frac{1}{e}}$.
- Running time is poly in the bit-size of the modulus and the degree.

RSA with random padding

Scenario:

- Two message m, m' are related: $m' = m + r \mod N$.
- We obtain the plain RSA encryptions with exponent $e = 3$
  $c = m^3 \mod N$ and $c' = (m + r)^3 = m^3 + 3m^2 r + 3mr^2 + r^3 \mod N$.

Exercise: Show that m can be efficiently computed from c, c' and r.

Question: What happens for *unknown but small r*?

- Question has applications for RSA with random padding R.
- Assume that we encrypt the same message M twice.
- Let the random padding be a k-bit string. Then

$$m = M \cdot 2^k + R,$$
$$m' = M \cdot 2^k + R'.$$

- Set $r = R' - R$, then $m' = m + r$ as before.

Attack on Random Padding RSA (Franklin, Reiter 96)

Theorem

Let $c = m^3 \bmod N$ and $c' = (m+r)^3 \bmod N$. Then m can be computed in time polynomial in $\log N$ provided that $|r| \leq N^{\frac{1}{9}}$.

Proof:

- Write $c' = (m+r)^3$ as
$$c' - m^3 - r^3 = 3m^2r + 3mr^2 = 3mr(m+r) \bmod N \,.$$
- Raising both sides to the 3rd power leads to
$$(c' - \underbrace{m^3}_{c} - r^3)^3 = 9\underbrace{m^3}_{c} r^3 \underbrace{(m+r)^3}_{c'} \bmod N.$$
- We obtain a monic polynomial $f(r)$ of degree 9.
- Coppersmith's method recovers r for $|r| \leq N^{\frac{1}{9}}$ in time poly in $\log N$.
- From c, c', r one can efficiently recover m (previous exercise).

Solving polynomial equations modulo divisors

Goal 2 Find small modular roots of polynomials

Given: integer M of unknown factorization, monic polynomial

$$f(x) = x^n + a_{n-1} + \ldots + a_1 x + a_0.$$

Find: all small roots x_0 with $f(x_0) = 0 \bmod b$ for some $b|M$.

Remarks:

- We do not know b, but it suffice to know a multiple M of b.
- Root $f(x_0) = 0 \bmod b$ usually give us factorization of M in b and $\frac{M}{b}$.
- Let $N = pq$. Consider the polynomial $f(x) = x \bmod p$.
- The roots of f are of the form $kp, k \in \mathbb{Z}$ and yield the factorization.
- We will first restrict to $f(x)$ of degree 1.

Coppersmith for divisors (1996)

Theorem Coppersmith for divisors

Let $\epsilon > 0$. For sufficiently large $M \in \mathbb{N}$ the following holds.
Let $f(x) = x + a$. Let b be a divisor of M with $b \geq M^{\beta}$, $0 < \beta \leq 1$. Then one can compute all x_0 with

$$f(x_0) = 0 \bmod b \text{ and } |x_0| \leq M^{\beta^2 - \epsilon}$$

in time polynomial in $\log M, \frac{1}{\beta}$ and $\frac{1}{\epsilon}$.

Proof:

- Choose suitable m. $\left(m = \lceil \frac{\beta^2}{\epsilon} \rceil\right)$
- Define the following collection of polynomials f_i of degree i.

$$\begin{aligned} f_i(x) &= M^{m-i} f^i(x) \quad \text{for } i = 0, \ldots, m \\ f_i(x) &= x^{i-m} f^m(x) \quad \text{for } i = m+1, \ldots, \tfrac{1}{\beta} m - 1 \end{aligned}$$

- If $f(x_0) = 0 \bmod b$ then $f_i(x) = 0 \bmod b^m$ for all i.
- Thus, $g(x) = \sum_i a_i f_i(x)$ fulfills condition 1 of Howgrave-Graham.

Coppersmith for divisors

Proof: (continued)

- Let X upper bound x_0. The coefficient vectors of $f(xX)$ form

$$B = \begin{pmatrix} M^m & & & & & & \\ - & M^{m-1}X & & & & & \\ & & \ddots & & & & \\ - & - & \dots & X^m & & & \\ & - & - & \dots & X^{m+1} & & \\ & & & & & \ddots & \\ & & & - & - & \dots & X^{\frac{1}{\beta}m-1} \end{pmatrix}.$$

- B spans a lattice L with $\mathrm{rank}(L) = \frac{1}{\beta}m$ and

$$\det(L) = \prod_{i=1}^{m} M \prod_{i=1}^{\frac{1}{\beta}m-1} X^i = M^{\frac{m(m+1)}{2}} X^{\frac{(\frac{1}{\beta}m-1)\frac{1}{\beta}m}{2}} \approx M^{\frac{m^2}{2}} X^{\frac{m^2}{2\beta^2}}.$$

- Each lattice vector corresponds to a coefficient vector of some $g(xX)$.
- Compute via LLL reduction a short vector $\mathbf{v}$ with

$$\|\mathbf{v}\| = \|g(xX)\| \le c^{\mathrm{rank}(L)} \cdot \det(L)^{\frac{1}{\mathrm{rank}(L)}}.$$

Coppersmith for divisors

Proof: (continued)

- Howgrave-Graham's second condition yields $\|g(xX)\| \leq \frac{b^m}{\sqrt{\mathrm{rank}(L)}}$.
- Omitting low-order terms, we simplify our condition to

$$\det(L) \leq b^m \cdot \dim(L).$$

- Using $b \geq M^\beta$, one obtains the more restrictive condition

$$\det(L) \leq M^{\beta m \mathrm{rank}(L)} \quad \Leftrightarrow M^{\frac{m^2}{2}} X^{\frac{m^2}{2\beta^2}} \leq M^{\beta m \cdot \frac{1}{\beta} m}$$
$$\Leftrightarrow M \cdot X^{\frac{1}{\beta^2}} \leq M^2 \qquad \Leftrightarrow X \leq M^{\beta^2}.$$

- Running time: LLL reduction on a rank $\frac{1}{\beta} m$ basis with entries of bit-size $\mathcal{O}(\frac{1}{\beta} m \log M)$. This is polynomial in $\frac{1}{\beta}, \log M$ and $m = \frac{\beta^2}{\epsilon}$.

Note: With additional tricks we can again omit the error term ϵ.

General form of Coppersmith's Theorem

Theorem Coppersmith

Let $\epsilon > 0$. For sufficiently large $M \in \mathbb{N}$ the following holds. Let $f(x)$ be a polynomial of degree n. Let b be a divisor of M with $b \geq M^{\beta}$, $0 < \beta \leq 1$. Then one can compute all x_0 with

$$f(x_0) = 0 \bmod b \text{ and } |x_0| \leq M^{\frac{\beta^2}{n}}$$

in time polynomial in $\log M, \frac{1}{\beta}$.

Factoring with high bits known (Coppersmith 96)

Scenario:

- Attacker knows the MSBs of p, e.g. via a side-channel attack.
- Vanstone-Zuccherato scheme: 264 of 512 bits represent identity.

Theorem

Let $N = pq$ with $p > q$. Let $\tilde{p}$ be a known approximation of p with $|p - \tilde{p}| \leq N^{\frac{1}{4}}$. Then N can be factored in time polynomial in $\log N$.

Proof:

- Define $f(x) = \tilde{p} + x$ with root $x_0 = p - \tilde{p} \bmod p$ and $|x_0| \leq N^{\frac{1}{4}}$.
- Since $p > q$ we have $p > N^{\frac{1}{2}}$. We set $\beta = \frac{1}{2}$.
- Coppersmith's Theorem: We can compute the root x_0 if
$$|x_0| \leq N^{\beta^2} = N^{\frac{1}{4}}.$$
- The root $x_0 = p - \tilde{p}$ gives the factorization $p = \tilde{p} + x_0$ and $q = \frac{N}{p}$.
- Our running time is polynomial in $\log N$.

Factoring with approximation of a multiple of p

Theorem

Let $N = pq$ with $p > q$. Let $\widetilde{kp}$ be a known approximation of kp with $|kp - \widetilde{kp}| \leq N^{\frac{1}{4}}$. Then N can be factored in time polynomial in $\log N$.

Proof: left as an exercise.

Scenario: Using bits of $d_p = d \bmod p - 1$ (Blömer, May 03)

- Attacker knows MSBs of $d_p = d \bmod p - 1$.
- We use a small encryption exponent e.

Using bits of $d_p = d \bmod p - 1$ (Blömer, May 03)

Theorem

Let $N = pq$, $p > q$ and $e = N^\alpha, 0 < \alpha \le \frac{1}{4}$. Let $\widetilde{d_p}$ be a known approximation of d_p with $|d_p - \widetilde{d_p}| \le N^{\frac{1}{4}-\alpha}$.
Then N can be factored in time polynomial in $\log N$.

Proof:

- We have $ed_p = 1 \bmod p - 1$ or equivalently $ed_p = 1 + k(p-1)$ with
$$k = \frac{ed_p - 1}{p-1} < e\frac{d_p}{p-1} < e.$$
- This implies $k < N^{\frac{1}{4}}$ and $q \nmid k$.
- We compute an approximation $\widetilde{kp} = e\widetilde{d_p} - 1$ satisfying
$$\begin{aligned} |kp - \widetilde{kp}| &= |ed_p - 1 + k - (e\widetilde{d_p} - 1)| \\ &= |e(d_p - \widetilde{d_p}) + k| \le N^\alpha N^{\frac{1}{4}-\alpha} + N^{\frac{1}{4}} \le 2N^{\frac{1}{4}} \end{aligned}$$
- With previous theorem: One of the values $\widetilde{kp} \pm N^{\frac{1}{4}}$ yields p, q.

Factoring $\equiv_{dp}$ Computing d (May 2004)

Theorem

Let $N = pq$ with p, q of equal bit-size. Assume we have an algorithm that computes d in polynomial time with $ed = 1 \bmod \phi(N)$, $ed < \phi(N)^2$. Then N can be factored in polynomial time.

Proof:

- We have $ed = 1 \bmod \phi(N)$, respectively $ed - 1 = k\phi(N)$.
- N is an approximation of $\phi(N)$ with $N - \phi(N) = p + q - 1 \leq 3N^{\frac{1}{2}}$.
- One of the values $N - \frac{i}{2}N^{\frac{1}{2}}$, $i = 0, \ldots, 5$ satisfies

$$\underbrace{N - \frac{i}{2}N^{\frac{1}{2}}}_{\widetilde{\phi(N)}} - \phi(N) \leq \tfrac{1}{2}N^{\frac{1}{2}}.$$

- Define $f(x) = \widetilde{\phi(N)} - x \bmod \phi(N)$ with root $x_0 = \widetilde{\phi(N)} - \phi(N)$,

$$x_0 \leq \tfrac{1}{2}N^{\frac{1}{2}} \leq \phi(N)^{\frac{1}{2}}.$$

Factoring $\equiv_{dp}$ Computing d (May 2004)

Proof: (continued)

- Let $M = ed - 1 = \phi(N)^\alpha$ for $\alpha < 2$. Define $b = \phi(N)$ and $\beta = \frac{1}{\alpha}$.
- Coppersmith's Theorem: We can compute x_0 as long as
$$|x_0| \le M^{\beta^2} \le (\phi(N))^{\frac{1}{4}} = \phi(N)^{\frac{1}{2}}.$$
- The value x_0 yields $\phi(N) = \widetilde{\phi(N)} - x_0$. The values of $\phi(N)$ and N together yield the factorization of N.
- Running time of our method is polynomial in $\log(M) \le 2\log N$.

Extensions to multivariate polynomials

Idea:

1. Construct k polynomials $g_1(x_1, \ldots, x_k), \ldots, g_k(x_1, \ldots, x_k)$ all sharing the same small roots over $\mathbb{Z}$.
2. Compute the common roots using resultants.

Problem: Does not work if $\gcd(g_i, g_j)$ is non-trivial.
(but usually works good in practice)

Some results using multivariate polynomials:

- Boneh-Durfee 99: Cryptanalysis of RSA with $d \leq N^{0.292}$.
- Jochemsz-May 07: Cryptanalysis of RSA with $d_p, d_q \leq N^{0.073}$.

The Digital Signature Algorithm (DSA)

Signature DSA

1. **Params:** public: $p, q \mid p-1$, $\alpha \in \mathbb{Z}_p^*$ with $\operatorname{ord}(\alpha) = q$, $\beta = \alpha^a \bmod p$
   private: $a \in \mathbb{Z}_q$
2. **Sign:** $\sigma(m) = (\gamma, \delta) = ((\alpha^r \bmod p) \bmod q, r^{-1}(m + a\gamma) \bmod q)$

Remarks:

- Knowledge of the randomization r immediately yields the secret a.
- If two messages are signed with the same r, then a can be efficiently computed. (Exercise)

Attack on DSA (Nguyen 1999)

Scenario:

- Attacker is allowed to query signature queries $\sigma_1, \ldots, \sigma_d$.
- For each σ_i the attacker gets ℓ LSBs of r, e.g., via side-channel.
- Example from practice: AT&T Crypto Lib always uses odd r.
- Let $r_i = r_i^{(m)} 2^\ell + r_i^{(\ell)}$ for $i = 1, \ldots, d$ with known $r_i^{(\ell)}$.
- Since $\delta_i = r_i^{-1}(x_i + a\gamma_i)$ we have

$$\begin{aligned} & a\gamma_i = \delta_i r_i - x_i = \delta_i (r_i^{(m)} 2^\ell + r_i^{(\ell)}) - x_i \mod q \\ \Rightarrow \quad & a \underbrace{\gamma_i \delta_i^{-1} 2^{-\ell}}_{t_i} = r_i^{(m)} + \underbrace{2^{-\ell} r_i^{(\ell)} - x_i \delta_i^{-1} 2^{-\ell}}_{\widetilde{at_i}} \mod q \end{aligned}$$

- Note that $\widetilde{at_i}$ is an approximation of at_i up to an error of

$$r_i^{(m)} = \frac{r_i - r_i^{(\ell)}}{2^\ell} < \frac{q}{2^\ell}.$$

Goal: Find the secret a using $t_1, \ldots, t_d$ and $\widetilde{at_1}, \ldots, \widetilde{at_d}$.

The Hidden Number Problem (Boneh, Venkatesan 96)

Definition Hidden Number Problem (HNP)

Given: prime q, $t_1, \ldots, t_d$ and $\widetilde{at_1}, \ldots, \widetilde{at_d}$ with

$$|(at_i \bmod q) - \widetilde{at_i}| \leq \frac{q}{2^d}.$$

Find: $a \in \mathbb{Z}_q$

Remark:

- We assume that the t_i are uniformly random chosen in $\mathbb{Z}_q$.
- If d and ℓ are sufficiently large then a is uniquely determined.

Lattice based solution of HNP (Boneh, Venkatesan)

Idea:

- Consider the lattice L spanned by the basis matrix

$$B = \begin{pmatrix} q & & & & 0 \\ & q & & & 0 \\ & & \ddots & & 0 \\ & & & q & 0 \\ t_1 & t_2 & \dots & t_d & \frac{1}{2^\ell} \end{pmatrix}.$$

- Obviously, $(at_1, at_2, \dots, at_d, \frac{a}{2^\ell}) \in L$ as well as

$$\mathbf{t} := (at_1 \bmod q, at_2 \bmod q, \dots, at_d \bmod q, \frac{a}{2^\ell}) \in L.$$

- From the vector $\mathbf{t}$ we can easily read of the desired secret a.
- We know a vector $\widetilde{\mathbf{t}} = (\widetilde{at_1}, \widetilde{at_2}, \dots, \widetilde{at_d}, 0)$ satisfying $\|\mathbf{t} - \widetilde{\mathbf{t}}\| = \|((at_1 \bmod q) - \widetilde{at_1}, \dots, (at_d \bmod q) - \widetilde{at_d}, \frac{a}{2^\ell})\| < \sqrt{d+1} \cdot \frac{q}{2^\ell}$.
- May hope that CVP in L with target $\widetilde{\mathbf{t}}$ yields $\mathbf{t}$ and thus a.

DSA attacks are practical

Theorem Nguyen

Every $\mathbf{u} \in L$ with $\|\mathbf{u} - \widetilde{\mathbf{t}}\| < \sqrt{d+1} \cdot \frac{q}{2^\ell}$ yields a with some probability which is constant for the parameters $d \sim \log q$ and $\ell \sim \log \log q$.

Remark:

- Evaluation of the probability for a 160-bit q yields an attack for
$$d = 100 \text{ and } \ell = 6.$$
- In practice even the following parameter choice suffices:
$$d = 100 \text{ and } \ell = 3.$$