Lecture 14

Thursday, 20 October 2022 10:45

LLL algorithm output satisfies $||v_1|| \le \left(2/\sqrt{4\delta-1}\right)^{(d-1)/2} \det(M)^{1/d}$

Typically we use LLL with $\delta = \frac{1}{2}$. This guarantees that the first, and shortest, vector satisfies $||v_1|| \quad 2^{\frac{d-1}{2}} \det(M)^{\frac{1}{d}}$ for M a $d \times d$-matrix.

Note: $\det(M) = \det(B)$ where B is LLL-reduced version of M.

Example:

$$M = \begin{pmatrix} X^2 & aX & 0 \\ 0 & X & a \\ 0 & 0 & n \end{pmatrix}, \quad \det(M) = X^3 n$$

$d = 3$

Theorem by Howgrave-Graham

Given some polynomial $g(x) \in \mathbb{Z}$, $\deg(g) = d-1$

If

1. $g(x_0) \equiv 0 \bmod b^k$

   for $b, k \in \mathbb{Z}_{>0}$ with $x_0 = X$ (same as $\mathbb{N} \setminus \{0\}$)

2. $||g(xX)|| \le \frac{b^k}{\sqrt{d}}$ for $||g(xX)|| = \sqrt{g_0^2 + g_1^2 X^2 + \dots + g_{d-1}^2 X^{2(d-1)}}$

then $g(x_0) = 0$ over $\mathbb{Z}$.

We can't solve polynomial equations modulo composites, but over $\mathbb{Z}$ is easy.

This shows why we could recover p from knowing the top $\frac{2}{3}$ (bits) of p.

LLL output gives $||v_1|| \le 2 \cdot X \sqrt[3]{n}$

We interpreted v_1 as coefficients of g (Q in large slide)

This makes $g(xX) = \sum_{i=1}^{3} a_i \text{ row } M(i)$ (i^{th} row of M)

for $a_i \in \mathbb{Z}$

Given that $p = a + x_0$, we have that $g(x_0) = 0$ for each of the rows mod p, so $b = p$ and $k = 1$. The LLL output means $||g(x)|| \le 2X\sqrt[3]{n}$, we need $||g(xX)|| \le \frac{p}{\sqrt{3}}$ for the second condition.

This is guaranteed to work for $2X\sqrt[3]{n} \le \frac{p}{\sqrt{3}}$ (can work better)

We know $p \simeq \sqrt{n}$ if p & q have same size. Thus it works if $X \le \frac{\sqrt{n}}{\sqrt[3]{n}\sqrt{3} \cdot 2} \approx \frac{\sqrt[6]{n}}{2\sqrt{3}} \approx \frac{\sqrt[3]{p}}{2\sqrt{3}}$

So we need to know at least the top $\frac{2}{3}$ (bits) of p.

We could have started with

$$M = \begin{pmatrix} X & a \\ 0 & n \end{pmatrix}, \quad \det(M) = nX, \quad d = 2$$

This works for $2^{\frac{2-1}{2}} (nX)^{\frac{1}{2}} \le \frac{p}{\sqrt{2}}$ (LLL-side; condition 2 side; for conditions 1 & 2 above, of the Howgrave-Graham theorem)

$$X^{\frac{1}{2}} \le \frac{p}{2\sqrt{n}} \approx \frac{1}{2}$$

→ no space for x_0.

Larger matrices (make LLL slower, but still polynomial time; makes factoring slower)

LLL output scales with $(\det(M))^{\frac{1}{d}}$, so avoid rows with n if we can.

$$M = \begin{pmatrix} X^3 & X^2 a & 0 & 0 \\ 0 & X^2 & Xa & 0 \\ 0 & 0 & X & a \\ 0 & 0 & 0 & n \end{pmatrix}$$

determinant computation using first row

$$\det(M) = X^3 \begin{vmatrix} X^2 & Xa & 0 \\ 0 & X & a \\ 0 & 0 & n \end{vmatrix} - X^2 a \begin{vmatrix} 0 & Xa & 0 \\ 0 & X & a \\ 0 & 0 & n \end{vmatrix} + 0 - 0 = X^3 \cdot X^3 \cdot n = X^6 n$$

This works for $2^{\frac{3}{2}} \sqrt[4]{X^6 n} \le \frac{p}{\sqrt{4}}$ (gets better bounds, despite increase in power of X)

constants have a minor impact

$$X^{\frac{3}{2}} \le \frac{p}{\sqrt{p}} = \sqrt{p}$$

$$X \le p^{\frac{1}{3}}$$

How about $d = 5$

$$M = \begin{pmatrix} X^4 & X^3 a & 0 & 0 & 0 \\ 0 & X^3 & X^2 a & 0 & 0 \\ 0 & 0 & X^2 & Xa & 0 \\ 0 & 0 & 0 & X & a \\ 0 & 0 & 0 & 0 & n \end{pmatrix}$$

$$\det(M) = X^{10} n$$

This works for

$$\sqrt[5]{X^{10} n} \le p$$

$$X^2 \le \frac{p}{p^{2/5}}$$

$$X = p^{\frac{3}{10}}$$

Coppersmith turns roots mod n into roots over $\mathbb{Z}$.

The Howgrave-Graham theorem shows why it works.

Build our g using LLL from polynomial $f(x)$ mod n (or mod p, in general mod b^k),

where we know a bound X on the desired root. We have seen $f(x) = a + x$ for a the top part of p and $f(x) = (a+x)^3 - c$ for a the top part of m.

Test yourself by finding f if bottom parts are known. Watch out to remove the large power of 2 from X, by defining f appropriately.

Hint: f is mod n or mod p, both of which are odd, hence 2^{-1} exists.

Let $\deg(f) = t$, then build matrix with rows $b^k, Xb^k, X^2 b^k, \dots, X^{t-1} b^k$, $f(xX), Xf(xX), \dots$, until Howgrave-Graham guarantees a solution (or try earlier).

Run LLL, take output (starting from first row), turn this into polynomial $g(x)$.

Note: the vectors are scaled by powers of X, so v_1 matches $g(xX)$, need to divide coefficient of x^i by X^i

$$g(x) = \sum_{i=0}^{d-1} v_i x^i / X^i$$

Clock group $x^2 + y^2 = 1$

This is working in $\mathbb{F}_{p^2}$ if $\sqrt{-1} \notin \mathbb{F}_p$ (else in $\mathbb{F}_p \times \mathbb{F}_p$) in a subgroup thereof.

There are index calculus attacks on the DLP in $\mathbb{F}_{p^n}$ that work similar to NFS (and are called NFS or function field sieve) and run in subexponential time