RSA cryptanalysis using the LLL lattice reduction algorithm

Meryem Cherkaoui Semmouni
Sidi Mohamed Ben Abdelah University, Fes, Morocco

Cedarcrypt, Cyprus, July 13, 2026

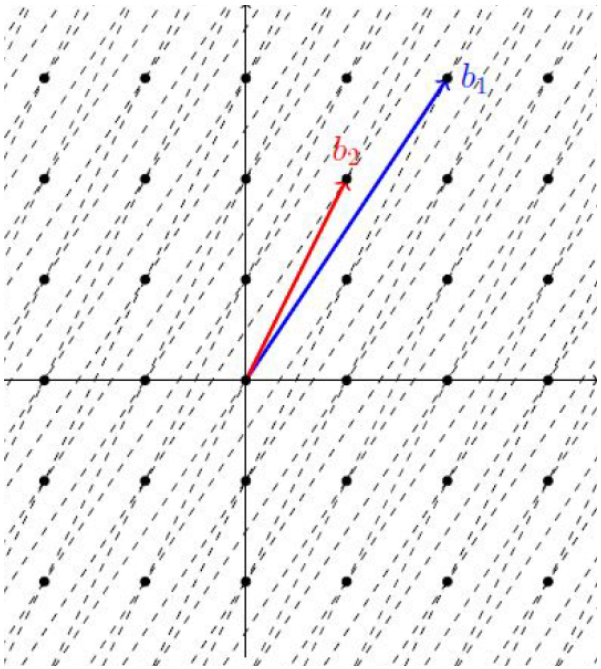
Lattices

A **Lattice** $\mathcal{L}$ generated by $B = (b_1, \cdots, b_d) \in (\mathbb{R}^n)^d$ is the set of all integer linear combinations of the linearly independent vectors $b_{1 \leq i \leq d}$:

Lattices

A **Lattice** $\mathcal{L}$ generated by $B = (b_1, \cdots, b_d) \in (\mathbb{R}^n)^d$ is the set of all integer linear combinations of the linearly independent vectors $b_{1 \leq i \leq d}$:

$$\mathcal{L} = \left\{ \sum_{i=1}^{d} \lambda_i b_i \;\middle|\; \lambda_i \in \mathbb{Z} \right\}.$$



The LLL Algorithm

The **Lenstra–Lenstra–Lovász (LLL)** algorithm is a polynomial-time lattice basis reduction algorithm.
Given a basis

$$\mathcal{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n),$$

LLL computes another basis of the same lattice containing shorter and more nearly orthogonal vectors :

- Gram–Schmidt orthogonalization to compute the projections $\mu_{i,j}$;
- size reduction:

$$|\mu_{i,j}| \leq \frac{1}{2};$$

- verification of the Lovász condition;
- exchange of vectors when the condition is not satisfied.

LLL

LLL Inequality

Let $\mathcal{L}$ be a lattice of dimension ω. The LLL algorithm returns, in polynomial time, a reduced basis $\{v_1, \cdots, v_\omega\}$ satisfying:

$$\|v_1\| \leq \|v_2\| \leq \cdots \leq \|v_i\| \leq 2^{\frac{\omega(\omega-1)}{4(\omega+1-i)}} \det(\mathcal{L})^{\frac{1}{\omega+1-i}}, \quad \text{for all } 1 \leq i \leq \omega.$$

Howgrave-Graham

Theorem

Let $M, X \in \mathbb{N}$ *and let* $f(x) = \sum_{i=0}^{d} a_i x^i \in \mathbb{Z}[x]$
Suppose $x_0 \in \mathbb{Z}$ *is a solution of* $f(x) \equiv 0 \pmod{M}$ *such that* $|x_0| \leq X$.

$$\textit{If} \quad \|f(xX)\| < \frac{M}{\sqrt{d+1}}, \quad \textit{then} \quad f(x_0) = 0 \quad \textit{over} \quad \textit{integers}.$$

Theorem

Soit $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ *un polynôme qui est une somme d'au plus* ω *monômes.*
$f\left(x_1^{(0)}, \ldots, x_n^{(0)}\right) \equiv 0 \pmod{N^m}$, $\left|x_1^{(0)}\right| < X_1, \ldots, \left|x_n^{(0)}\right| < X_n$,
$\|f(x_1X_1, \ldots, x_nX_n)\| < \frac{N^m}{\sqrt{\omega}}$
Alors $f\left(x_1^{(0)}, \ldots, x_n^{(0)}\right) = 0$ *sur les entiers.*

Coppersmith's Algorithm

Univariate Modular Polynomial

- Basic setup: a univariate monic polynomial

$$f(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_2x^2 + a_1x + a_0,$$

over $\mathbb{Z}[x]$ with degree $d > 1$ and a modulus M of unknown factorization.

- Goal – To find "small roots" x_0 such that $|x_0| < B$, for a suitable bound B and

$$f(x_0) \equiv 0 \pmod{M}.$$

Coppersmith proposed a method, where $B = M^{1/d}$.

Coppersmith's Algorithm

The Central Problem

Suppose $\exists$ at least one solution x_0 to

$$f(x) \equiv 0 \pmod{M}$$

and that

$$|x_0| \leq M^{1/d}.$$

Coppersmith's Algorithm

Coefficients are small enough

- Find roots over $\mathbb{Z}$:
  - Get roots over $\mathbb{R}$: Newton's method
  - Round approximation of the roots to nearest integer x_0
  - Check whether $f(x_0) = 0$ over $\mathbb{Z}$
- Go to mod M

Coppersmith's Idea

Coefficients are not small

- Build $g(x) \in \mathbb{Z}[x]$ from $f(x)$ such that

$$f(x_0) \equiv 0 \pmod{M} \quad \Longrightarrow \quad g(x_0) = 0 \text{ over } \mathbb{Z}$$

Coppersmith Lattice Basis Construction

Let $g_i(x) = Mx^i$, for $0 \leq i < d$, be d polynomials that have the root x_0 modulo M. The lattice basis B is constructed from the coefficient vectors of :

$$g_0(xX), g_1(xX), \ldots, g_{d-1}(xX), f(xX)$$

Equivalently, each coefficient of x^i is multiplied by X^i. Thus, the basis B of the lattice $\mathcal{L}$ is defined as follows:

$$B = \begin{bmatrix} M & 0 & 0 & \cdots & 0 & 0 \\ 0 & MX & 0 & \cdots & 0 & 0 \\ 0 & 0 & MX^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & MX^{d-1} & 0 \\ a_0 & a_1X & a_2X^2 & \cdots & a_{d-1}X^{d-1} & X^d \end{bmatrix}.$$

From the lattice to a polynomial

Idea

The rows of the matrix B correspond to the coefficient vectors of

$$g_0(xX), g_1(xX), \ldots, g_{d-1}(xX), f(xX).$$

Therefore, any vector of the lattice $\mathcal{L}(B)$ corresponds to an integer linear combination of these polynomials.

More precisely, if

$$v = (c_0, c_1X, c_2X^2, \ldots, c_dX^d) \in \mathcal{L}(B),$$

then v corresponds to the polynomial

$$g(x) = c_0 + c_1x + c_2x^2 + \cdots + c_dx^d.$$

Why does $g(x_0)$ vanish modulo M?

Assume that x_0 is a small root of $f(x)$ modulo M, i.e.

$$f(x_0) \equiv 0 \pmod{M} \quad \text{and} \quad |x_0| \leq X.$$

For the auxiliary polynomials

$$g_i(x) = Mx^i, \qquad 0 \leq i < d,$$

we also have

$$g_i(x_0) = Mx_0^i \equiv 0 \pmod{M}.$$

Thus, every integer linear combination of

$$g_0(x), g_1(x), \ldots, g_{d-1}(x), f(x)$$

also vanishes at x_0 modulo M.

$$g(x_0) \equiv 0 \pmod{M}.$$

Applying LLL

We apply the LLL algorithm to the lattice basis B. LLL returns a reduced basis whose first vector is short.
Let

$$v = (c_0, c_1X, c_2X^2, \ldots, c_dX^d)$$

be the first vector of the LLL-reduced basis.

This vector defines the polynomial

$$g(x) = c_0 + c_1x + c_2x^2 + \cdots + c_dx^d.$$

Since v is short, the norm of $g(xX)$ is small:

$$\|g(xX)\| < \frac{M}{\sqrt{d+1}}.$$

From modular root to integer root

We know that the polynomial $g(x)$ satisfies

$$g(x_0) \equiv 0 \pmod{M}.$$

Moreover, LLL gives a short vector, so

$$\|g(xX)\| < \frac{M}{\sqrt{d+1}}.$$

By Howgrave-Graham's theorem, this implies

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

Therefore, the modular equation

$$f(x_0) \equiv 0 \pmod{M}$$

is transformed into an ordinary integer equation

$$g(x_0) = 0.$$

Recovering the small root

After LLL, we obtain a polynomial $g(x)$ such that

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

Then we proceed as follows:

1. Solve the equation
$$g(x) = 0$$
over the integers.
2. Keep only the integer roots satisfying
$$|x_0| \leq X.$$
3. Verify each candidate by checking
$$f(x_0) \equiv 0 \pmod{M}.$$

The valid candidates are the small modular roots of $f(x)$ modulo M.

Coppersmith's Method: Summary

1. Start with a monic polynomial $f(x) \in \mathbb{Z}[x]$ of degree d and a modulus M.
2. Choose a bound X such that the root satisfies
$$|x_0| \leq X.$$
3. Construct the lattice basis B from the coefficient vectors of
$$g_0(xX), g_1(xX), \ldots, g_{d-1}(xX), f(xX),$$
where $g_i(x) = Mx^i$.
4. Apply LLL to obtain a short vector and construct the polynomial $g(x)$.
5. Solve $g(x) = 0$ over $\mathbb{Z}$.
6. Output the roots x_0 satisfying
$$|x_0| \leq X \quad \text{and} \quad f(x_0) \equiv 0 \pmod{M}.$$

Important theorems and Background

Theorem

Suppose given a basis B, and $g(x)$ be the polynomial corresponding to the first vector in the LLL-reduced basis for $\mathcal{L}$. If

$$X < \frac{M^{\frac{2}{d(d+1)}}}{\sqrt{2}\,(d+1)^{1/d}},$$

then any root x_0 of $f(x)$ modulo M such that $|x_0| \leq X$ satisfies

$$g(x_0) = 0 \quad \text{in } \mathbb{Z}.$$

Fixed Padding RSA: Construction of the Polynomial

- The padding is added by the encryption process.
- If the padding scheme is fixed and known, the attacker knows the structure of the plaintext.
- The plaintext can be written as

$$m = A + K,$$

where:

- A is the known fixed padding part;
- K is the unknown secret part, for example a 128-bit AES key.

Fixed Padding RSA: Polynomial Construction

RSA encryption with public exponent $e = 3$ gives

$$c \equiv m^3 \pmod{N}.$$

Since the plaintext has the form

$$m = A + K,$$

we get

$$c \equiv (A + K)^3 \pmod{N}.$$

Therefore,

$$(A + K)^3 - c \equiv 0 \pmod{N}.$$

We define the polynomial $f(x) = (A + x)^3 - c$
Then the unknown secret K satisfies $f(K) \equiv 0 \pmod{N}$
Since K is small, Coppersmith's method can be applied to recover K.

Fixed Padding RSA: Coppersmith Setup

We have obtained the polynomial

$$f(x) = (A + x)^3 - c.$$

The unknown secret K satisfies

$$f(K) \equiv 0 \pmod{N}.$$

Since K is a 128-bit value, we know that

$$0 \leq K < 2^{128}.$$

Thus, we set the bound

$$X = 2^{128}.$$

The problem becomes:

$$\text{Find a small root } K \text{ of } f(x) \text{ modulo } N \quad \textit{suchthat} \quad |K| \leq X$$

This is exactly the type of problem solved by Coppersmith's method.

Fixed Padding RSA: Polynomial Expansion

Recall that

$$f(x) = (A + x)^3 - c.$$

By expanding, we get

$$f(x) = x^3 + 3Ax^2 + 3A^2x + (A^3 - c).$$

So the polynomial has degree

$$d = 3.$$

It can be written as

$$f(x) = a_0 + a_1x + a_2x^2 + x^3,$$

where

$$a_0 = A^3 - c, \qquad a_1 = 3A^2, \qquad a_2 = 3A.$$

Since $f(K) \equiv 0 \pmod{N}$, the value K is a small modular root of $f(x)$.

Fixed Padding RSA: Auxiliary Polynomials

Since the degree of $f(x)$ is

$$d = 3,$$

we construct the auxiliary polynomials

$$g_i(x) = Nx^i, \qquad 0 \leq i < 3.$$

Thus,

$$g_0(x) = N, \qquad g_1(x) = Nx, \qquad g_2(x) = Nx^2.$$

These polynomials vanish modulo N for every integer x:

$$g_i(x) \equiv 0 \pmod{N}.$$

Together with $f(x)$, we use

$$g_0(x), \quad g_1(x), \quad g_2(x), \quad f(x)$$

to construct the lattice basis.

Fixed Padding RSA: Scaling by X

Since the unknown root satisfies

$$|K| \leq X,$$

we construct the lattice using the polynomials evaluated at xX.

We consider

$$g_0(xX), \quad g_1(xX), \quad g_2(xX), \quad f(xX).$$

The auxiliary polynomials become

$$g_0(xX) = N,$$
$$g_1(xX) = NXx,$$
$$g_2(xX) = NX^2x^2.$$

For the polynomial f, we have

$$f(xX) = (A + xX)^3 - c.$$

Fixed Padding RSA: Coefficient Vectors

We have

$$f(x) = x^3 + 3Ax^2 + 3A^2x + (A^3 - c).$$

Therefore,

$$f(xX) = X^3x^3 + 3AX^2x^2 + 3A^2Xx + (A^3 - c).$$

The coefficient vector of $f(xX)$ is

$$(A^3 - c,\ 3A^2X,\ 3AX^2,\ X^3).$$

Similarly, the coefficient vectors of the auxiliary polynomials are

$$g_0(xX): \quad (N,\ 0,\ 0,\ 0),$$
$$g_1(xX): \quad (0,\ NX,\ 0,\ 0),$$
$$g_2(xX): \quad (0,\ 0,\ NX^2,\ 0).$$

Fixed Padding RSA: Lattice Basis

The lattice basis B is constructed from the coefficient vectors of

$$g_0(xX), \quad g_1(xX), \quad g_2(xX), \quad f(xX).$$

Thus, we obtain the matrix

$$B = \begin{bmatrix} N & 0 & 0 & 0 \\ 0 & NX & 0 & 0 \\ 0 & 0 & NX^2 & 0 \\ A^3 - c & 3A^2X & 3AX^2 & X^3 \end{bmatrix}.$$

Then we apply the LLL algorithm to B.
The first short vector of the LLL-reduced basis gives a new polynomial

$$h(x)$$

such that the small root K satisfies

$$h(K) = 0 \quad \text{over } \mathbb{Z}.$$

Fixed Padding RSA: Applying LLL

We have constructed the lattice basis

$$B = \begin{bmatrix} N & 0 & 0 & 0 \\ 0 & NX & 0 & 0 \\ 0 & 0 & NX^2 & 0 \\ A^3 - c & 3A^2X & 3AX^2 & X^3 \end{bmatrix}.$$

We apply the LLL algorithm to this basis:

$$B_{\text{red}} = \text{LLL}(B).$$

Let the first vector of the LLL-reduced basis be

$$v = (b_0,\ b_1X,\ b_2X^2,\ b_3X^3).$$

This vector corresponds to the polynomial

$$h(x) = b_0 + b_1x + b_2x^2 + b_3x^3.$$

Fixed Padding RSA: Why does $h(K) = 0$?

The vector obtained by LLL corresponds to an integer linear combination of

$$N, \quad Nx, \quad Nx^2, \quad f(x).$$

Hence, the polynomial $h(x)$ satisfies

$$h(K) \equiv 0 \pmod{N}.$$

Since LLL returns a short vector, the polynomial $h(x)$ has small coefficients.

By Howgrave-Graham's theorem, if

$$\|h(xX)\| < \frac{N}{\sqrt{d+1}},$$

then the modular relation becomes an equality over the integers:

$$h(K) = 0 \quad \text{over } \mathbb{Z}.$$

Fixed Padding RSA: Recovering K

After LLL, we solve $h(x) = 0$ over the integers.
Let the integer roots of $h(x)$ be

$$r_1, r_2, \ldots, r_s.$$

We keep only the roots satisfying the bound

$$0 \leq r_i < X.$$

Then we verify each candidate by checking

$$f(r_i) \equiv 0 \pmod{N}.$$

The valid candidate is the secret value:

$$K = r_i.$$

Therefore, the attacker recovers the unknown AES key K.

Boneh-Durfee

BONEH-DURFEE

Private Exponent Relation

$$e \cdot d \equiv 1 \pmod{\varphi(N)}$$

Why is a Small Private Exponent Dangerous?

In RSA, the private exponent satisfies

$$ed \equiv 1 \pmod{\varphi(N)}.$$

Therefore, there exists an integer k such that

$$ed = k\varphi(N) + 1.$$

If the private exponent is small, we write

$$d < N^{\delta}.$$

Then the unknown integer k is also small, because

$$k \approx \frac{ed}{\varphi(N)}.$$

For standard RSA sizes, e and $\varphi(N)$ are of the same order, so

$$k < N^{\delta}.$$

Introducing k

$$e \cdot d \equiv 1 \pmod{\varphi(N)}$$

$$\Longrightarrow e \cdot d = k \cdot \varphi(N) + 1$$

Modulo e

$$e \cdot d = k \cdot \varphi(N) + 1$$

$$\Longrightarrow e \cdot d - k\varphi(N) - 1 = 0$$

$$\Longrightarrow k \cdot \varphi(N) + 1 \equiv 0 \pmod{e}$$

Using Euler's Totient

$$\varphi(N) = (p-1)(q-1)$$

$$\varphi(N) = N + 1 - p - q$$

$$k \cdot \varphi(N) + 1 \equiv 0 \pmod{e}$$

$$\Longrightarrow k(N + 1 - p - q) + 1 \equiv 0 \pmod{e}$$

From RSA to a Bivariate Modular Polynomial

We have obtained

$$k(N+1-p-q)+1 \equiv 0 \pmod{e}.$$

Let

$$A = N+1, \qquad x = k, \qquad y = -(p+q).$$

Then

$$N+1-p-q = A+y.$$

Thus, the RSA equation becomes

$$x(A+y)+1 \equiv 0 \pmod{e}.$$

We define the bivariate polynomial

$$f(x,y) = x(A+y)+1.$$

Therefore, the unknown root satisfies

$$f(x_0,y_0) \equiv 0 \pmod{e},$$

Bounds on the Small Root

The unknown root is

$$(x_0, y_0) = (k, -(p+q)).$$

Since we assume $d < N^\delta$, we also have $k < N^\delta$. we choose $X = N^\delta$.

Also, since $N = pq$ and p, q are balanced primes,

$$p \approx q \approx \sqrt{N}.$$

Hence,

$$p + q \approx 2\sqrt{N}.$$

So we choose

$$Y = N^{1/2}.$$

Thus, the small root satisfies

$$|x_0| < X = N^\delta, \qquad |y_0| < Y = N^{1/2}.$$

Connection with Coppersmith's Method

Coppersmith's method finds small roots of modular polynomial equations.
In Boneh-Durfee, we have the bivariate modular equation

$$f(x, y) = x(A + y) + 1 \equiv 0 \pmod{e}.$$

The goal is to recover the small root

$$(x_0, y_0) = (k, -(p + q)).$$

To apply Coppersmith's method, we build shifted polynomials from

$$f(x, y).$$

These shifts vanish at (x_0, y_0) modulo powers of e.
Then we construct a lattice from their coefficient vectors and apply LLL.
LLL gives short vectors, which correspond to polynomials that vanish at

$$(x_0, y_0) \quad \textit{over} \quad \textit{the} \quad \textit{integers}$$

.

Boneh-Durfee Shifts: From Polynomials to Matrix Rows

We consider the bivariate polynomial

$$f(x,y) = 1 + x(A+y) = 1 + Ax + xy.$$

For $m = 2$ and $t = 1$, the shifts are:

$$g_{i,j}(x,y) = x^j e^{m-i} f(x,y)^i, \quad \textit{where} \quad 0 \le i \le m, \qquad 0 \le j \le m-i.$$

$$e^2, \quad xe^2, \quad ef, \quad x^2e^2, \quad xef, \quad f^2,$$

together with the y-shifts:

$$h_{i,j}(x,y) = y^j f(x,y)^i e^{m-i}, \quad \textit{where} \quad 0 \le i \le m, \qquad 1 \le j \le t.$$

$$ye^2, \quad yef, \quad yf^2.$$

Boneh-Durfee Shifts: From Polynomials to Matrix Rows

To construct the lattice, we scale the variables using the bounds:

$$x \leftarrow Xx, \qquad y \leftarrow Yy.$$

Thus,

$$f(Xx, Yy) = 1 + AXx + XYxy.$$

The rows of the basis matrix are the coefficient vectors of the scaled shifts.

Boneh-Durfee x-Shifts

Using $f(Xx, Yy) = 1 + AXx + XYxy$ we obtain:

$$e^2 \longrightarrow e^2,$$

$$xe^2 \longrightarrow e^2Xx,$$

$$ef \longrightarrow e + eAXx + eXYxy,$$

$$x^2e^2 \longrightarrow e^2X^2x^2,$$

$$xef \longrightarrow eXx + eAX^2x^2 + eX^2Yx^2y,$$

$$f^2 \longrightarrow 1 + 2AXx + 2XYxy + A^2X^2x^2 + 2AX^2Yx^2y + X^2Y^2x^2y^2.$$

Boneh-Durfee y-Shifts

The y-shifts are obtained by multiplying by Yy.

$$ye^2 \longrightarrow e^2Yy,$$

$$yef \longrightarrow eYy + eAXYxy + eXY^2xy^2,$$

$$yf^2 \longrightarrow Yy+2AXYxy+2XY^2xy^2+A^2X^2Yx^2y+2AX^2Y^2x^2y^2+X^2Y^3x^2y^3.$$

These coefficient vectors form the last three rows of the basis matrix.

Boneh-Durfee Matrix: Monomial Order

We choose the following monomial order for the columns:

$$1, \quad x, \quad xy, \quad x^2, \quad x^2y, \quad x^2y^2, \quad y, \quad xy^2, \quad x^2y^3.$$

Each row of the matrix is obtained by writing one scaled shift as a coefficient vector with respect to this monomial order.

For example,

$$ef(Xx, Yy) = e + eAXx + eXYxy$$

gives the row

$$(e,\ eAX,\ eXY,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0).$$

Similarly,

$$xef(Xx, Yy) = eXx + eAX^2x^2 + eX^2Yx^2y$$

gives the row

$$(0,\ eX,\ 0,\ eAX^2,\ eX^2Y,\ 0,\ 0,\ 0,\ 0).$$

Bivariate Howgrave-Graham

Let $g(x, y)$ be a bivariate polynomial with at most w monomials. If:

$$g(x_0, y_0) \equiv 0 \pmod{e^m}$$

where:

$$|x_0| < X \quad \text{and} \quad |y_0| < Y$$

and:

$$\|g(xX, yY)\| < \frac{e^m}{\sqrt{d+1}},$$

then:

$$g(x_0, y_0) = 0 \quad \text{over the integers.}$$

Boneh-Durfee Basis Matrix for $m = 2,\ t = 1$

	1	x	xy	x^2	x^2y	x^2y^2	y	xy^2	x^2y^3
e^2	e^2	0	0	0	0	0	0	0	0
xe^2	0	e^2X	0	0	0	0	0	0	0
fe	e	eAX	eXY	0	0	0	0	0	0
x^2e^2	0	0	0	e^2X^2	0	0	0	0	0
xfe	0	eX	0	eAX^2	eX^2Y	0	0	0	0
f^2	1	$2AX$	$2XY$	A^2X^2	$2AX^2Y$	X^2Y^2	0	0	0
ye^2	0	0	0	0	0	0	e^2Y	0	0
yfe	0	0	$eAXY$	0	0	0	eY	eXY^2	0
yf^2	0	0	$2AXY$	0	A^2X^2Y	$2AX^2Y^2$	Y	$2XY^2$	X^2Y^3

Why Does the Determinant Matter?

After building the Boneh-Durfee lattice B, we apply LLL.
LLL gives a short vector v satisfying approximately

$$\|v\| \lesssim 2^{\frac{n-1}{4}} \det(B)^{1/n},$$

where n is the lattice dimension.

This short vector corresponds to a polynomial $g(x, y)$.
To use Howgrave-Graham's theorem, we need

$$\|g(Xx, Yy)\| < e^m.$$

Therefore, a sufficient condition is

$$2^{\frac{n-1}{4}} \det(B)^{1/n} < e^m.$$

Ignoring the LLL constant, the main condition becomes

$$\det(B) < e^{mn}.$$

Removing the Damaging y-Shifts

	1	x	xy	x^2	x^2y	x^2y^2	y	xy^2	x^2y^3
e^2	e^2	0	0	0	0	0	0	0	0
xe^2	0	e^2X	0	0	0	0	0	0	0
fe	e	eAX	eXY	0	0	0	0	0	0
x^2e^2	0	0	0	e^2X^2	0	0	0	0	0
xfe	0	eX	0	eAX^2	eX^2Y	0	0	0	0
f^2	1	$2AX$	$2XY$	A^2X^2	$2AX^2Y$	X^2Y^2	0	0	0
yf^2	0	0	$2AXY$	0	A^2X^2Y	$2AX^2Y^2$	Y	$2XY^2$	X^2Y^3

After removing the damaging y-shifts coefficient vectors

Herrmann and May

Herrmann and May: Unravelled Linearization

$$f(x, y) = 1 + xy + Ax \pmod{e}$$

$$u = xy + 1$$

Unravelled Linearization

$$f(x, y) = 1 + xy + Ax$$

$$u = xy + 1$$

$$f(x, u) = u + Ax$$

Herrmann-May Shifts

We use the linearized polynomial

$$f(x,u) = u + Ax.$$

- $g'_{i,j}(x,u) = x^j e^{m-i} f(x,u)^i, \quad where \;\; 0 \leq i \leq m, \qquad 0 \leq j \leq m-i.$
- $h'_{i,j}(x,u,y) = y^j e^{m-i} f(x,u)^i, \quad where \;\; 0 \leq i \leq m, \qquad 1 \leq j \leq t.$

For this reduced example, we take

$$m = 2, \qquad t = 1.$$

Herrmann-May Basis Matrix

	1	x	u	x^2	ux	u^2	u^2y
e^2	e^2	0	0	0	0	0	0
xe^2	0	e^2X	0	0	0	0	0
$\bar{f}e$	0	eAX	eU	0	0	0	0
x^2e^2	0	0	0	e^2X^2	0	0	0
$x\bar{f}e$	0	0	0	eAX^2	eUX	0	0
$\bar{f}^2$	0	0	0	A^2X^2	$2AUX$	U^2	0
$y\bar{f}^2$	0	$-A^2X$	$-2AU$	0	A^2UX	$2AU^2$	U^2Y

After Building the Herrmann-May Matrix

- We apply the LLL algorithm:

$$B_{\text{red}} = \text{LLL}(B).$$

- LLL returns a reduced basis whose first vectors are short.
- Each short vector corresponds to a polynomial in the variables : $x, \quad u, \quad y$.
- The goal is to obtain short polynomials that vanish at the small root : (x_0, u_0, y_0) such as :

$$H(x_0, u_0, y_0) = 0 \quad \text{over } \mathbb{Z}.$$

From LLL Vectors to Polynomials

The columns of the matrix correspond to the monomial order

$$1, \quad x, \quad u, \quad x^2, \quad ux, \quad u^2, \quad u^2y.$$

Let v be a short vector obtained after LLL:

$$v = (c_0,\ c_1X,\ c_2U,\ c_3X^2,\ c_4UX,\ c_5U^2,\ c_6U^2Y).$$

Then this vector corresponds to the polynomial

$$H(x,u,y) = c_0 + c_1x + c_2u + c_3x^2 + c_4ux + c_5u^2 + c_6u^2y.$$

Because the vector is short, the polynomial H has small coefficients.

Recovering the Small Root

After LLL, we usually take the first short polynomials:

$$H_1(x, u, y), \qquad H_2(x, u, y), \qquad \dots$$

They satisfy

$$H_i(x_0, u_0, y_0) = 0 \quad \text{over } \mathbb{Z}.$$

We also have the unravelled relation

$$u = xy + 1.$$

Therefore, to recover the root, we solve the system

$$\begin{cases} H_1(x, u, y) = 0, \\ H_2(x, u, y) = 0, \\ u - xy - 1 = 0. \end{cases}$$

This can be done using resultants or a Gröbner basis.

Resultants

Eliminate variables step by step.

- eliminate u between

$$H_1(x, u, y) = 0 \quad and \quad u - xy - 1 = 0.$$

- Then eliminate y to obtain a univariate polynomial in x:

$$R(x) = 0.$$

- The integer roots of $R(x)$ give candidates for x_0.

Grobner Basis

- We consider the ideal

$$I = \langle H_1(x, u, y),\ H_2(x, u, y),\ u - xy - 1 \rangle.$$

- Then we compute a Gröbner basis of I.
- the Gröbner basis may contain a univariate polynomial: P(x)=0
- Solving $P(x) = 0$ gives the possible values of x_0.
- After finding x_0, we recover $u_0 = x_0 y_0 + 1$.

Using the Determinant to Choose the Bounds

LLL gives a short vector v satisfying approximately

$$\|v\| \lesssim 2^{\frac{n-1}{4}} \det(B)^{1/n},$$

where n is the lattice dimension.
This vector corresponds to a polynomial $H(x, u, y)$.
To apply Howgrave-Graham's theorem, we need

$$\|H(Xx, Uu, Yy)\| < e^m.$$

Thus, we require

$$2^{\frac{n-1}{4}} \det(B)^{1/n} < e^m.$$

Neglecting the LLL constant, the main condition is

$$\det(B) < e^{mn}.$$

This inequality determines whether the chosen bounds X, U, Y are small enough for the attack to succeed.

Final Choice of the Bounds

From the RSA relation, the small root is

$$(x_0, y_0) = (k, -(p+q)).$$

The bounds are chosen as

$$X = N^{\delta}, \qquad Y = N^{1/2}.$$

The determinant condition

$$\det(B) < e^{mn}$$

checks whether these bounds are small enough for LLL to produce a useful short polynomial.
The Boneh-Durfee analysis shows that this condition can be satisfied when

$$\delta < 0.292.$$

Therefore, the attack can recover the private exponent when

$$d < N^{0.292}.$$

ECDSA cryptanalysis using LLL

LLL can also be used to attack ECDSA when some bits of the random nonces used by the signer to generate signatures are leaked, since this partial information may allow the private key to be recovered.

Question?