LATTICE BASIS REDUCTION

5. CRYPTOGRAPHIC APPLICATIONS

Alfred Menezes
cryptography101.ca

October 2025

Outline

1. LLL applications
2. ECDSA signature generation
3. Leakage: two signatures
4. Leakage: many signatures
5. Example

Cryptanalytic application of LLL

LLL is a very powerful cryptanalytic tool, and has been used to cryptanalyze many public-key cryptosystems, including:

- Knapsack public-key encryption schemes (subset sum problem)
- RSA with partial information
- DSA/ECDSA with partial information
- Factoring integers of the form $p^r q$
- Number Field Sieve (NFS) for factoring integers and computing discrete logarithms.

General methodology

To solve a cryptanalytic problem:

1. Define a lattice L in such a way that the solution to the cryptanalytic problem is encoded in a nonzero lattice vector v of unusually small length.
2. Run the LLL algorithm to find a short basis B for L.
3. Check whether v is in B.

Side-channel attacks on ECDSA

- In the ECDSA signature scheme, a signer has a long-term private key a.
- The signer signs a message using a and also a per-message secret k.
  - Both a and k must be generated *uniformly at random* from the private key space (integers in an interval $[1, n-1]$).
  - Also, a fresh secret k must be generated for each message that is signed.
- In practice, information about some per-messages secrets might be leaked during the signing process via a side-channel attack.
  - Poor implementation, weak random number generator, timing analysis, electromagnetic radiation analysis, etc.
- This side-channel information can sometimes be used to compute the long-term private key a.

ECDSA: Domain parameters and key generation

ECDSA domain parameters:

- Elliptic curve E defined over $\mathbb{Z}_p$ (e.g., P-256).
- $n = \#E(\mathbb{Z}_p)$, where n is prime.
- A generator $P \in E(\mathbb{Z}_p)$.
- A collision-resistant hash function H whose output has the same bitlength as n (e.g., H = SHA256).

ECDSA key generation: Alice does

1. Select $a \in_R [1, n-1]$ and compute $A = aP$.
2. Alice's public key is A; her private key is a.

Note: Computing a from A is an instance of the ECDLP

ECDSA: Signature generation

ECDSA signature generation: To sign a message $M \in \{0,1\}^*$, Alice does:

1. Compute $m = H(M)$ and interpret m as an integer.
2. Select a per-message secret $k \in_R [1, n-1]$.
3. Compute $R = kP$. Let $r = x(R) \bmod n$, and check that $r \neq 0$.
   [$x(R)$ is the x-coordinate of R.]
4. Compute $s = k^{-1}(m + ar) \bmod n$, and check that $s \neq 0$.
5. Alice's signature on M is (r, s).

Leakage: two signatures (1)

- We'll first consider the scenario where partial information is leaked about the per-message secrets for *two* signed messages.
- Recall: an ECDSA signature on a message M is (r, s), where $s = k^{-1}(m + ar) \bmod n$, $k \in_R [1, n-1]$ is the per-message secret, $a \in_R [1, n-1]$ is the long-term private key, and $m = H(M)$.
- Thus, knowledge of a signed message (M, r, s) yields a linear congruence in two unknowns: $ks \equiv m + ar \pmod{n}$.
- Now, suppose that an adversary obtains two signed messages (M_1, r_1, s_1) and (M_2, r_2, s_2). This yields two linear congruences: $k_1 s_1 \equiv m_1 + ar_1 \pmod{n}$ and $k_2 s_2 \equiv m_2 + ar_2 \pmod{n}$.
- Solving for a and equating yields the congruence $k_1 s_1 r_2 - m_1 r_2 \equiv k_2 s_2 r_1 - m_2 r_1 \pmod{n}$.
- Rearranging gives the linear congruence $k_1 - (s_1 r_2)^{-1} s_2 r_1 k_2 + (s_1 r_2)^{-1}(m_2 r_1 - m_1 r_2) \equiv 0 \pmod{n}$, where the only quantities that are not known to the adversary are k_1 and k_2.

Leakage: two signatures (2)

- *Let's assume that* $k_1, k_2 \leq K$, *where* $K \ll n$ *is a parameter we will specify later.*
- The congruence $k_1 - (s_1 r_2)^{-1} s_2 r_1 k_2 + (s_1 r_2)^{-1}(m_2 r_1 - m_1 r_2) \equiv 0 \pmod{n}$ can be written as $k_1 + tk_2 + u \equiv 0 \pmod{n}$ where $t = -(s_1 r_2)^{-1} s_2 r_1 \bmod n$ and $u = (s_1 r_2)^{-1}(m_2 r_1 - m_1 r_2) \bmod n$.
- This congruence can then be written as an equation $k_1 + tk_2 + nx + u = 0$ where $x \in \mathbb{Z}$. The adversary can compute t and u, and knows n.
- Now, consider the integer lattice L with basis vectors $(n,0,0)$, $(t,1,0)$, and $(u,0,K)$.
- The basis matrix for L is $B = \begin{bmatrix} n & t & u \\ 0 & 1 & 0 \\ 0 & 0 & K \end{bmatrix}$, so the volume of L is $\mathrm{vol}(L) = nK$.

Leakage: two signatures (3)

- Note that $v = B[x, k_2, 1]^T = [xn + tk_2 + u, k_2, K] = [-k_1, k_2, K]$ is a lattice vector of length $\|v\| = \sqrt{k_1^2 + k_2^2 + K^2} \leq \sqrt{3K^2} = \sqrt{3}K$.
- The Gaussian heuristic asserts that $\lambda_1(L) \approx \sqrt{3/2\pi e}\,(\mathrm{vol}(L))^{1/3} \approx (nK)^{1/3}$.
- So, if $\|v\| \ll (nK)^{1/3}$, we can expect that v is a shortest nonzero vector in L.
  - This condition is $\sqrt{3}K \ll (nK)^{1/3}$, or $K \ll \sqrt{n}$.
- In this event, the adversary can (likely) determine v by solving SVP in L.
  - Knowledge of v immediately reveals k_1 and k_2, and then the long-term private key a can be efficiently computed from the signing equation.

Leakage: many signatures (1)

- Suppose now that the adversary has obtained N signed messages (M_i, r_i, s_i).
- Suppose also that the adversary somehow obtains a few least significant bits of each per-message secret k_i.
  - More precisely, let $k_i = \alpha_i 2^\ell + \beta_i$, where $\beta_i < 2^\ell$.
  - We're assuming that the adversary knows β_i but not α_i.
  - Let $K = \lceil n/2^\ell \rceil$, so $\alpha_i < K$.

- For each $1 \le i \le N-1$, eliminate a from the signing equations for M_i and M_N to obtain $k_i + t_i' k_N + u_i' \equiv 0 \pmod{n}$.
- Next, replace each k_i by $\alpha_i 2^\ell + \beta_i$ to obtain $(\alpha_i 2^\ell + \beta_i) + t_i'(\alpha_N 2^\ell + \beta_N) + u_i' \equiv 0 \pmod{n}$.
- Finally, multiply both sides by $2^{-\ell} \bmod n$, simplify, and eliminate the "mod n" to obtain equations $\alpha_i + t_i \alpha_N + u_i + x_i n = 0$.

Leakage: many signatures (2)

- We have $N-1$ equations $\alpha_i + t_i\alpha_N + u_i + x_i n = 0$, where the unknown quantities are the α_i (for $1 \le i \le N$) and the x_i (for $1 \le i \le N-1$).
- Now, consider the $(N+1)$-dimensional integer lattice $L \subseteq \mathbb{Z}^{N+1}$ with basis matrix

$$B = \begin{bmatrix} n & 0 & \cdots & 0 & t_1 & u_1 \\ 0 & n & \cdots & 0 & t_2 & u_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & n & t_{N-1} & u_{N-1} \\ 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & K \end{bmatrix}_{(N+1)\times(N+1)}$$

- The volume of the lattice is $\mathrm{vol}(L) = n^{N-1}K$.
- Note that $v = B[x_1, x_2, \ldots, x_{N-1}, \alpha_N, 1]^T = (-\alpha_1, -\alpha_2, \ldots, -\alpha_{N-1}, \alpha_N, K) \in L$ is a lattice vector of length $\|v\| = \sqrt{\alpha_1^2 + \cdots + \alpha_N^2 + K^2} \le \sqrt{N+1}\,K \approx \sqrt{N}K$.

Leakage: many signatures (3)

- The Gaussian heuristic asserts that

$$\begin{aligned}\lambda_1(L) &\approx \sqrt{(N+1)/2\pi e}\,(\mathrm{vol}(L))^{1/(N+1)} \\ &\approx N^{1/2}(n^{N-1}K)^{1/(N+1)} \\ &= N^{1/2}n^{(N-1)/(N+1)}K^{1/(N+1)}.\end{aligned}$$

- We desire that $\|v\|$ is significantly shorter than the above estimate for $\lambda_1(L)$, so $N^{1/2}K \ll N^{1/2}n^{(N-1)/(N+1)}K^{1/(N+1)}$, which simplifies to $K^N \ll n^{N-1}$.
- Let $t = \lceil \log_2 n \rceil$, whence $\log_2 K \approx t - \ell$.
- The inequality $K^N \ll n^{N-1}$ simplifies to $N(t-\ell) \ll Nt - t$, which yields $N \gg t/\ell$.
- If $N \gg t/\ell$, the adversary can (likely) determine v by solving SVP in L.
  - Knowledge of v immediately reveals the k_i, and then the long-term private key a can be efficiently computed.

Leakage: many signatures (4)

- **Summary**: If the adversary learns the ℓ least significant bits of each per-message secret k_i for $N \gg t/\ell$ signatures (where t is the bitlength of n), then the adversary can likely determine the k_i (and thereafter the long-term private key a).
- Experiments have shown that the attack is successful for ℓ as small as 2.
- The attack can be adapted to situations where the adversary learns a few most significant bits of per-message secrets, or a few middle bits of per-message secrets.
- The attack has been effective in breaking poor implementations of ECDSA in practice, including in Bitcoin applications.

Example: ECDSA attack (1)

- Select $n = 1299709$ (21-bit), $N = 6$, $K = 30000$.
- We'll suppose that each per-message secret k_i is less than K.
  - This is (roughly) equivalent to knowing that the 6 most significant bits of each k_i are 0.
- The five congruences are $k_i + t_i k_6 + u_i \equiv 0 \pmod{n}$, $1 \le i \le 5$.
- Suppose that $k_1 = 11977$, $k_2 = 13859$, $k_3 = 24557$, $k_4 = 8359$, $k_5 = 11865$, $k_6 = 3790$.
  - The bitlengths of the k_i are 14, 14, 15, 14, 14, and 12, respectively.
- Select random t_i: $t_1 = 542359$, $t_2 = 1033924$, $t_3 = 1200045$, $t_4 = 703576$, $t_5 = 977601$.
- Compute $u_i = -k_i - t_i k_6 \bmod n$:
  - $u_1 = 587051$, $u_2 = 36816$, $u_3 = 786393$, $u_4 = 441469$, $u_5 = 350704$.
- Given n, t_i and u_i, the adversary's task is to determine the k_i.

Example: ECDSA attack (2)

- The associated lattice is $L = L(B) \subseteq \mathbb{Z}^7$ where

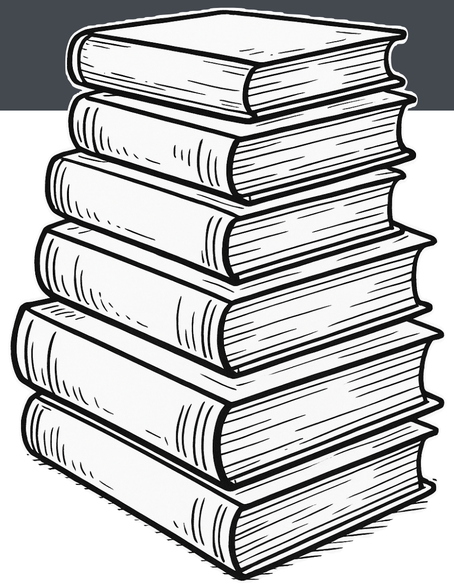
$$B = \begin{bmatrix} n & 0 & 0 & 0 & 0 & t_1 & u_1 \\ 0 & n & 0 & 0 & 0 & t_2 & u_2 \\ 0 & 0 & n & 0 & 0 & t_3 & u_3 \\ 0 & 0 & 0 & n & 0 & t_4 & u_4 \\ 0 & 0 & 0 & 0 & n & t_5 & u_5 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & K \end{bmatrix}_{7\times 7} = \begin{bmatrix} 1299709 & 0 & 0 & 0 & 0 & 542359 & 587051 \\ 0 & 1299709 & 0 & 0 & 0 & 1033924 & 36816 \\ 0 & 0 & 1299709 & 0 & 0 & 1200045 & 786393 \\ 0 & 0 & 0 & 1299709 & 0 & 703576 & 441469 \\ 0 & 0 & 0 & 0 & 1299709 & 977601 & 350704 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 30000 \end{bmatrix}.$$

- The LLL algorithm produces the following lattice basis:
  (-11977, -13859, -24557, -8359, -11865, 3790, 30000) , (-21678, -24134, 58135, -30130, -18940, -47952, 0)
  (-47506, 42635, 47880, -59539, -15567, 43551, 30000), (-49163, 20976, -65451, -14745, 57733, -51258, -30000),
  (19124, 21634, 61374, 15710, 35769, -28367, 90000), (16861, -70562, -7745, -99833, 68811, 3482, 0),
  (77910, 34861, -34327, -61281, -78499, -76662, -30000).

- The $-k_i$ appear in the first five components of the first basis vector.

Check: $A = aP$.

Other LLL applications



1. **Phong Nguyen** and **Jacques Stern**,
   "The two faces of lattices in cryptology",
   CaLC 2001, https://doi.org/10.1007/3-540-44670-2_12

2. **Gabrielle De Micheli** and **Nadia Heninger**,
   "Recovering cryptographic keys from partial information, by example",
   IACR Communications in Cryptology, 2024, https://cic.iacr.org/p/1/1/28

3. **Joachim Breitner** and **Nadia Heninger**,
   "Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies",
   Financial Crypto 2019, https://eprint.iacr.org/2019/023