Information Security Lab
Autumn Semester 2022
Module 1, Week 2 – Cryptanalysis of ECDSA

Kenny Paterson (@kennyog)

Applied Cryptography Group

https://appliedcrypto.ethz.ch/

Overview of today's lectures

- ECDSA recap from last week

- Breaking ECDSA with a single known nonce and with repeated nonces

- Breaking ECDSA with partially known nonces

- Introducing lattices and lattice reduction

- From breaking ECDSA to CVP

- From CVP to SVP via Kannan embedding

- Putting it all together

ECDSA Recap

ECDSA Recap

Parameters: (E, p, n, q, h, P, H) defining a curve E over field F_p with $n = q \cdot h$ points, subgroup of prime order q and generator P of order q; H is a hash function, e.g. SHA-256 (here we assume output of H is at least bit-size of q).

KeyGen:

> Set $Q = [x]P$ where x is uniformly random from $\{1,...,q\text{-}1\}$.
>
> Output verification key: **Q**; signing key: **x**.

Sign: Inputs (d, m)   // d is private key; m is the message to be signed

> $h = \text{bits2int}(H(m)) \bmod q$.   // take len($q$) MSBs of $H(m)$, cast to BigInt, reduce mod q.
>
> Do:
>
> 1. Select k uniformly at random from $\{1,...,q\text{-}1\}$.   // k is called the *nonce*
>
> 2. Compute $r = \text{x-coord}([k]P) \bmod q$. // $[k]P$ is a point on E; its x-coord is in F_p; we consider that as an integer and reduce mod q.
>
> 3. Compute $s = k^{-1}(h + xr) \bmod q$.
>
> Until $r \neq 0$ and $s \neq 0$.   // works first try w.h.p.
>
> Output (r, s).

ECDSA Recap

Verify: Inputs (Q, m, (r,s)) // Q is verification key; m is message; (r, s) is claimed signature.

1. check that $1 \leq r \leq q-1$ and $1 \leq s \leq q-1$.
2. compute $w = s^{-1}$ mod q.
3. compute $h =$ bits2int($H(m)$) mod q.
4. compute $u_1 = w \cdot h$ mod q and $u_2 = w \cdot r$ mod q.
5. compute $Z = [u_1]P + [u_2]Q$.
6. If (x-coord(Z) mod $q == r$) then output 1 else output 0.

Correctness:

Suppose (r,s) is a signature for message m under key Q. Then:

$$Z = [u_1]P + [u_2]Q = [s^{-1}h]P + [s^{-1}r]Q = [s^{-1}(h + xr)]P = [k]P.$$

Here we used $s = k^{-1}(h + xr)$ mod q from the signing algorithm to obtain $s^{-1}(h + xr) = k$ mod q.

Recalling that $r =$ x-coord($[k]P$) mod q completes the argument.

5

Simple Attacks on ECDSA

Breaking ECDSA with A Single Known Nonce

- Suppose we have an ECDSA signature (r,s) for message m in which the nonce k is known in its entirety.

- Recall the signing equation:

$$s = k^{-1}(h + xr) \bmod q.$$

- Rearranging gives:

$$x = r^{-1}(ks - h) \bmod q.$$

- Here, $h = \text{bits2int}(H(m)) \bmod q$, so depends only on m.

- Hence recovering x, the signing key, is trivial.

- You will implement this attack as a warm-up in the lab.

Breaking ECDSA with Repeated Nonces

- Suppose we have two ECDSA signatures (r_1,s_1) for message m_1 and (r_2,s_2) for message m_2.

- Suppose we also know that the same value k was used in both cases.

- For example, maybe the developer did not realise that k needs to be a fresh, random value for each use, or used a broken RNG.

- Sony Playstation 3 fail: https://www.bbc.com/news/technology-12116051

- Rearranging the signing equations gives the pair of linear equations:

$$xr_1 - ks_1 = h_1 \bmod q \quad \text{and} \quad xr_2 - ks_2 = h_2 \bmod q$$

  in unknowns x and k.

- Solve using linear algebra mod q to recover both unknowns.

- Explicitly: $x = (h_1 s_2 - h_2 s_1) \cdot (r_2 s_1 - r_1 s_2)^{-1} \bmod q$.

- You will implement this attack as a second warm-up in the lab.

Breaking ECDSA with Partially Known Nonces

Breaking ECDSA with Partially Known Nonces

- The preceding examples show that the security of ECDSA is very sensitive to how nonces k are chosen.

- How sensitive exactly?

- What if the attacker could learn just a few bits of k?

- Such information might be available via a side-channel attack.

- Examples:

  - Brumley and Tuveri, "Remote Timing Attacks are Still Practical", ESORICS 2011 and https://eprint.iacr.org/2011/232.pdf

  - Moghimi et al., "TPM-FAIL: TPM Meets Lattice and Timing Attacks", USENIX 2020 https://www.usenix.org/conference/usenixsecurity20/presentation/moghimi-tpm

  - Both papers observe leakage of cases when MSBs of k are zero due to faster execution of $[k]P$ during the signing algorithm: a timing side-channel observable by a "remote" attacker.

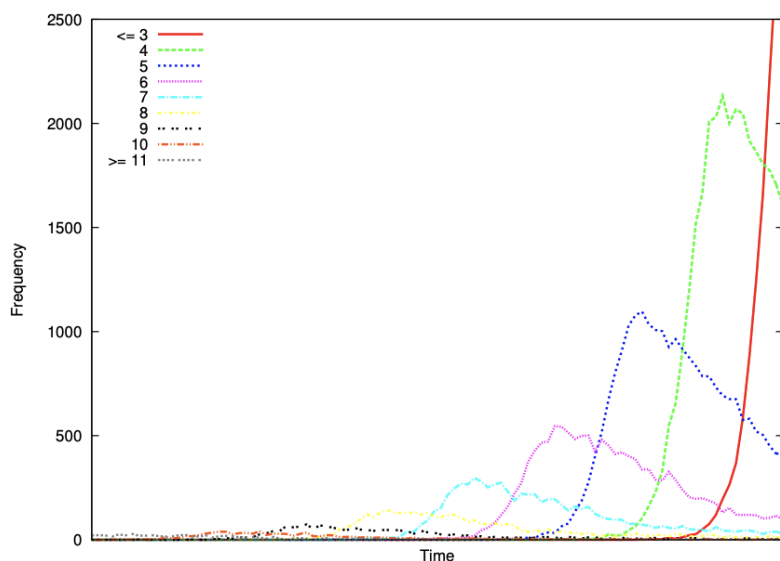 - Other, recent work observes partial leakage of k via cache-based side-channel attacks (a local attacker model).

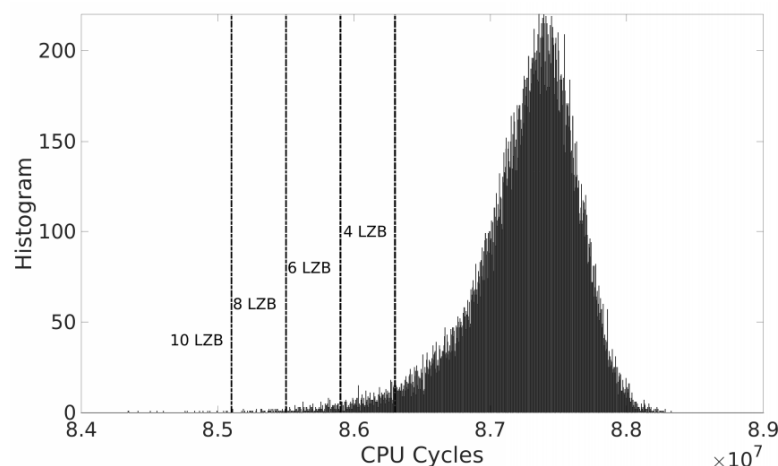Fig. 4. Dependency between number of leading zero bits and wall clock execution time of the signature operation.

Figure 2: Histogram of ECDSA (NIST-256p) signature generation timings on the STMicroelectronics TPM as measured on a Core i7-8650U machine for 40,000 observations.

Brumley-Tuveri, 2011
OpenSSL ECDSA signature generation

Moghimi et al., 2020
STMicroelectronics TPM ECDSA signature generation

Table 2: Discovered vulnerabilities in OpenSSL, LibreSSL, and BoringSSL and whether they are patched ✓ as of October 2019, currently being patched 🔧, or unpatched ✗. Exploiting the side channel can be easy ●, medium ◐, or hard ○. The number of leaked bits (Nonce Leakage) indicates the complexity of a full key recovery.

Vulnerability		OpenSSL	LibreSSL	BoringSSL	Nonce Leakage	SC	Comments
Generate: (V1) Small k (top)		EC✗	EC✗	–	Topmost 0-limbs of k	●	Leaks in several subsequent steps
Exp. / Scalar Mult.	(V2) k-padding resize	DSA✓EC✓	DSA✗EC✗	–	Topmost 0-bits of k	●	CVE-2018-0734 and CVE-2018-0735
	(V3) consttime-swap	DSA✓EC✓	DSA✗EC✗	–	same as (V2)	◐	Already known
	(V4) Downgrade	DSA✓	–	–	same as (V2) + [24]	●	Introduced while fixing (V2)
	(V5) k-padding (top)	DSA✗EC✗	DSA✗EC✗	–	same as (V2)	○	Leaks in BN_add and BN_is_bit_set. SGX attack shown in Appendix B.
	(V6) Buffer conversion	EC✓	–	–	Topmost 0-bytes of k	○	
	(V7) Point addition	EC🔧	–	EC✓	All 0-windows of k	○	
Invert	(V8) Euclid BN_div	DSA✓	DSA✗	–	Topmost bit of k	●	Leaks via resize, similar to (V2)
	(V9) Euclid negation	DSA✓	DSA✗	–	Topmost 0-bit of $kinv$	●	Leaks via conditional negation
Multiply: (V10) Small k^{-1} (top)		–	EC✓	–	Topmost 0-limbs of $kinv$	◐	

Weiser et al., "Big Numbers - Big Troubles: Systematically Analyzing Nonce Leakage in (EC)DSA Implementations", USENIX 2020, https://www.usenix.org/system/files/sec20-weiser.pdf

Breaking ECDSA with Partially Known Nonces

- You will implement a synthetic version of these attacks in the lab to recover signing keys.

- The remainder of these lectures will describe how you will do this, in a sequence of steps:

  - Reducing to Closest Vector Problem (CVP) in a lattice.

  - Reducing CVP to Shortest Vector Problem (SVP).

  - Solving SVP using the LLL algorithm (as a black box).

- The original ideas go back to: Howgrave-Graham and Smart, Lattice Attacks on Digital Signature Schemes. *Des. Codes Cryptography*, 23:283–290, 2001.

Lattices and Lattice Reduction

Lattices

Definition (full-rank lattice)

Let $\{\underline{b}_1, \underline{b}_2, ..., \underline{b}_n\}$ be n linearly independent vectors in $\mathbb{R}^n$.

Then the lattice generated by $\{\underline{b}_1, \underline{b}_2, ..., \underline{b}_n\}$ is the set:

$$L := \{ \sum_{i=1}^{n} l_i\, \underline{b}_i : l_i \in \mathbb{Z} \}$$

of **integer** linear combinations of the $\underline{b}_i$.

- If we allow real-valued linear combinations, we just get $L = \mathbb{R}^n$.
- Example: $n=2$; $\underline{b}_1 = (1,0)$; $\underline{b}_2 = (0,1)$, then $L = \mathbb{Z} \times \mathbb{Z}$.
- Example: $n=2$; $\underline{b}_1 = (12.3, \pi)$; $\underline{b}_2 = (-5, e)$, then $L = ??$.
- In general, the sums, differences etc, of lattice vectors are also lattice vectors.
- So a lattice is a *discrete subgroup* of $\mathbb{R}^n$.

Lattice Bases

<u>Definition (basis matrix of a full-rank lattice):</u>

A basis matrix B of a lattice $L \subset \mathbb{R}^n$ is an $n \times n$ matrix formed by taking the rows to be basis vectors $\underline{b}_i$. Then

$$L = \{ \underline{x}B : \underline{x} \in \mathbb{Z}^n \}$$

- In cryptanalysis applications, our matrix entries will usually be integers.
- Two different, full-rank matrices B, B' can generate the same lattice.

<u>Fact:</u>

Two different matrices B, B' generate the same lattice L if and only if B' = $U\,B$ where U is an $n \times n$ matrix with integer entries and determinant ±1.

<u>Definition (determinant of a full-rank lattice):</u>

The determinant of a full-rank lattice L is the absolute value of the determinant of any basis matrix B for the lattice.

<u>Definition (norm of a vector):</u>

For $\underline{v} \in \mathbb{R}^n$, $\|\underline{v}\|$ denotes the Euclidean norm of $\underline{v}$, i.e.

$$\|\underline{v}\| = \left(\sum_{i=1}^{n} \boldsymbol{v_i^2}\right)^{1/2}.$$

<u>Definition (successive minima of a lattice):</u>

Let $L \subset \mathbb{R}^n$ be a full rank lattice. The successive minima of L are the values $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ such that:

For $1 \leq i \leq n$, λ_i is the smallest real value such that there exist i linearly independent vectors $\underline{v}_1, \ldots, \underline{v}_i$ with $\|\underline{v}_j\| \leq \lambda_i$ for $1 \leq j \leq i$.

<u>Special case:</u>

λ_1 is the length of a **shortest** (in terms of Euclidean norm) non-zero vector in L.

17

There are many bounds on the size of the successive minima, particularly for λ_1.

The <mark>Gaussian Heuristic</mark>

Let $L \subset \mathbb{R}^n$ be a "random" full rank lattice. Then

$$\lambda_1 \approx (n/2\pi e)^{1/2} \cdot \det(L)^{1/n}.$$

NB: "*random lattice*" is not formally defined, but the ones we use in cryptanalysis can often be assumed to behave in this way.

SVP and CVP

Let $L \subset \mathbb{R}^n$ be a "random" full rank lattice, and let $\underline{w} \in \mathbb{R}^n$.

Then the **Shortest Vector Problem (SVP)** is to find $\underline{v} \in L$ such that $\| \underline{v} \| = \lambda_1$.

The **Closest Vector Problem (CVP)** for $\underline{w}$ is to find $\underline{v} \in L$ such that $\| \underline{v} - \underline{w} \|$ is as small as possible.

- SVP and CVP are known to be hard problems in general.

- For example, CVP can be shown to be NP-hard by relating it to subset-sum.

- SVP and CVP problems in high-dimensional lattices can be used to **construct** cryptographic schemes, e.g. public key encryption, signatures.

- However, SVP and CVP may be easy if the dimension is small and/or the lattice is presented in a "nice" way.

- Then lattices can be used as a tool for **cryptanalysis**.

Lattice Reduction

- Let $L \subset \mathbb{R}^n$ be a full rank lattice, represented by some basis matrix B.

- **Lattice reduction** refers to the process of producing a new basis matrix B' for L satisfying certain special properties.

- In particular, the rows of B' (whose linear combinations define L) are "somewhat orthogonal" and the norms of the first rows of B' are relatively short.

- The **Lenstra, Lenstra, Lovasz (LLL) algorithm** is a deterministic algorithm which performs lattice reduction.

- LLL essentially performs "iterative rounded Gram-Schmidt orthogonalization".

- Algorithmic details can be found in Chapter 17 of Galbraith's book "Mathematics of Public Key Cryptography" available for free at:
  https://www.math.auckland.ac.nz/~sgal018/crypto-book/crypto-book.html

Lattice Reduction – LLL and BKZ

- LLL runs in time (and space) polynomial in n and $\max_i \| \underline{b}_i \|^2$ and produces a basis matrix B' in which the first row $\underline{b}'_1$ satisfies:

$$\| \underline{b}'_1 \| \leq 2^{(n-1)/2} \lambda_1 .$$

- In other words, LLL *approximately* solves the SVP for lattice L (and more).

- In practice, LLL often *exactly* solves SVP.

- For large dimensions n, LLL is superseded by the BKZ algorithm (which uses LLL as a subroutine to solve SVP on sub-lattices).

- Due to its importance in cryptography, lattice reduction is a major area of on-going research, with many improvements to BKZ in recent years.

- In the lab, we will be using **fpylll**, a state-of-the-art lattice reduction package for Python.

- Nice intro at: https://martinralbrecht.wordpress.com/2016/04/03/fpylll/

From ECDSA to CVP

- Suppose we are given a signature (r,s) on message $m.$

- Suppose also that the L MSBs of the corresponding nonce k are known.

  - Let N be the bit-length of $q.$

  - Then k lies in the interval [$a2^{N-L}$, $(a+1)2^{N-L}-1$] for some known a (coming from MSBs of k).

  - The mid-point of this interval is $a2^{N-L} + 2^{N-L-1}.$

  - So let's write $k = a2^{N-L} + 2^{N-L-1} + e$ where $0 \leq |e| \leq 2^{N-L-1}.$

- Rearranging the signing equation $s = k^{-1}(h + xr)$ mod q yields:

$$(rs^{-1})x = k - hs^{-1} \bmod q$$

- Set $t = rs^{-1}$ mod q and $z = hs^{-1}$ mod q (z is an integer between 0 and q-1).

- Then we have:

$$tx = k - z \bmod q.$$

- We have: $\quad tx = k - z \bmod q.$
- Now:

$$k = a2^{N-L} + 2^{N-L-1} + e,$$

so

$$k - z = u + e$$

where

$$u = a2^{N-L} + 2^{N-L-1} - z$$

is a known integer that can be computed from the L MSBs of k and $z = hs^{-1} \bmod q.$

- Moreover, e is bounded by: $0 \leq |e| \leq 2^{N-L-1}.$
- So we finally arrive at: $tx = u + e \bmod q,$ and hence:

$$tx = u + e + l \cdot q \quad \text{for some } l$$

- Since $u = a2^{N-L} + 2^{N-L-1} - z,$ we see that u lies between $2^{N-L-1} - q$ and $a2^{N-L} + 2^{N-L-1}.$
- We really only care about values mod $q,$ so we can assume (by adding multiples of q as needed) that u is *centred*, i.e. $-q/2 < u < q/2.$

- So far: from signature (r,s) on message m and L MSBs of k we get:

$$tx = u + e + l \cdot q \quad \text{for some } l.$$

- Here, x is our target, t is known, u is known, e is small but otherwise unknown and l is unknown.

- We get one such equation for each of n signatures (r_i, s_i) on messages m_i:

$$t_i x = u_i + e_i + l_i \cdot q$$

- Alternatively, we can write:

$$t_i x = u_i + e_i \mod q \qquad \text{(where } e_i \text{ is small)}$$

meaning that u_i is a good approximation to $t_i x \mod q$.

- In this second form, the problem of recovering x from n distinct equations with uniformly random t_i is called the **Hidden Number Problem (HNP).**

- A similar translation can be done when the LSBs of the k_i are known, or in fact any set of contiguous bits.

A Formal Result:

Let $t_1, \ldots, t_n$ be uniformly random values in F_q, let x be non-zero in F_q. Suppose we are given n samples of the form (t_i, u_i) where u_i is known to be a good approximation to $t_i x \bmod q$ (i.e. $t_i x = u_i + e_i \bmod q$ with $0 \leq |e_i| \leq q/2^{L+1}$).

Suppose $n = 2\log_2(q)^{1/2}$ and $L = \log_2(q)^{1/2} + \log_2\log_2(q)$.

Then one can recover x in polynomial time.

- For a proof, see Theorem 21.7.9 and Corollary 21.7.10 of Galbraith's book.

- The proof is based on properties of LLL and the Babai nearest plane algorithm for solving CVP.

- Guarantee required of $|e_i|$ is slightly stronger here than in our formulation.

- In practice, we can get away with much smaller n and L and still get an attack that works.

26

Consider the lattice $L \subset \mathbb{R}^{n+1}$ with basis matrix B given by:

$$
\begin{vmatrix}
q & 0 & 0 & \ldots & 0 & 0 \\
0 & q & 0 & \ldots & 0 & 0 \\
. & & & & . & . \\
. & . & . & \ldots & . & . \\
. & & & & & \\
0 & 0 & 0 & \ldots & q & 0 \\
t_1 & t_2 & t_3 & & t_n & 1/2^{L+1}
\end{vmatrix}
$$

Define $\underline{u} = (u_1, u_2, \ldots u_n, 0) \in \mathbb{R}^{n+1}$ where, recall, $t_i x = u_i + e_i + l_i \cdot q$ with e_i small.

Claim: There exists a vector $\underline{v} \in L$ such that

$$\| \underline{u} - \underline{v} \| < (n+1)^{1/2} \cdot 2^{N-L-1}.$$

Proof of claim:

We can write

$$t_i\, x = u_i + e_i + l_i \cdot q$$

where $|e_i| \leq 2^{N-L-1}$, and l_i is some (unknown) integer.

Now define $\underline{v} \in L$ via:

$$\underline{v} = (-l_1,\ -l_2\ ,\ldots,\ -l_n,\ x) \cdot B.$$

(Recall that x is unknown, so $\underline{v}$ is also unknown at this point.)

$$\underline{v} = (-l_1,\ -l_2\ ,\ldots,\ -l_n,\ x\ ) \cdot B = (-l_1 q + t_1 x,\ -l_2 q + t_2 x,\ \ldots,\ -l_n q + t_n x,\ x/2^{L+1})$$

$$= (u_1 + e_1,\ u_2 + e_2,\ \ldots,\ u_n + e_n,\ x/2^{L+1})$$

Hence $\underline{v} - \underline{u} = (e_1,\ e_2,\ \ldots,\ e_n,\ x/2^{L+1})$ and the result follows on noting that each entry in $\underline{v} - \underline{u}$ is bounded in absolute value by 2^{N-L-1} (for the last entry, note that $x < q < 2^N$).

28

From ECDSA to CVP

Implications:

- We have constructed a lattice $L \subset \mathbb{R}^{n+1}$ and vector $\underline{u}$ from public information and shown that $\underline{u}$ is "somewhat close" to a point $\underline{v}$ in L.

- Moreover, finding $\underline{v}$ allows us to find the ECDSA private key x (just by inspecting the final coordinate of $\underline{v}$).

- We can hope that $\underline{v}$ is actually **the** solution to the CVP for $\underline{u}$.

- For some parameter ranges, one can show that this is indeed the case.

- So if we have a CVP solver, we can apply it here and hope to extract the private key x.

- So how do we solve CVP?

From CVP to SVP: Kannan's Embedding Technique

From CVP to SVP: Kannan's Embedding Technique

- There are multiple ways to solve CVP using an SVP solver: the Babai nearest plane algorithm, Babai rounding, Kannan's embedding technique, enumeration approaches.

- We will describe only Kannan embedding here, as it is nice and simple.

- Babai nearest plane and Babai rounding are also simple, and have provable guarantees.

- Your generic LLL library may allow you to solve CVP directly, but it's good to have a sense of what could be happening underneath!

Let B be a basis matrix for a lattice $L \subset \mathbb{R}^n$ with rows $\underline{b}_i$.

Let $\underline{w} \in \mathbb{R}^n$ be a vector for which we wish to solve CVP.

A solution to the CVP corresponds to integers $l_1, \ldots, l_n$ such that:

$$\underline{w} \approx l_1 \underline{b}_1 + \ldots + l_n \underline{b}_n.$$

Define $\underline{f} = \underline{w} - (l_1 \underline{b}_1 + \ldots + l_n \underline{b}_n)$.

Key observation: $\| \underline{f} \|$ is small.

So we try to define a new lattice L' which contains $\underline{f}$ – hopefully then $\underline{f}$ will be output as a result of running an SVP solver on L'.

- Consider the lattice $L' \subset \mathbb{R}^{n+1}$ with basis matrix B' whose rows are:

$$(\underline{b_1}, 0), (\underline{b_2}, 0), \ldots, (\underline{b_n}, 0), (\underline{w}, M)$$

 where, recall, $\underline{w}$ is the input to the CVP.

- Here M is a constant, to be determined.

- Now consider the linear combination of rows with coefficients:

$$(-l_1, \ldots, -l_n, 1)$$

- It is easy to check that this yields the vector $(\underline{f}, M)$, which should be short.

- So we might be able to solve CVP on input $\underline{w}$ for lattice L by solving SVP on lattice L' to find $(\underline{f}, M)$ and then setting $\underline{v} = \underline{w} - \underline{f}$.

33

Lemma:

Let $L \subset \mathbb{R}^n$ be a full rank lattice with shortest non-zero vector of length λ_1. Let $\underline{w} \in \mathbb{R}^n$ and let $\underline{v}$ be a closest lattice vector to $\underline{w}$. Define $\underline{f} = \underline{w} - \underline{v}$. Suppose that $\|\underline{f}\| \leq \lambda_1/2$ and let $M = \|\underline{f}\|$. Then $(\underline{f}, M)$ **is** a shortest vector in the lattice $L' \subset \mathbb{R}^{n+1}$ in Kannan's embedding technique.

Proof: see Galbraith's book, Lemma 18.3.2.

Interpretation: if the target vector $\underline{w}$ is very close to the lattice and we have a good guess M for the distance, then Kannan's embedding technique does reduce the problem of solving CVP to that of solving SVP.

Problems: maybe $\underline{w}$ is not close to the lattice; LLL and related algorithms only approximately solve SVP; maybe target $(\underline{f}, M)$ is short but **not** a shortest vector in L'.

Solution: in practice, this approach works quite well, but we may need to examine several vectors in the reduced basis to find target $(\underline{f}, M)$ or perform *enumeration*.

Putting It All Together

Putting It All Together

- We have seen how to translate the problem of recovering x from partial information about the nonces in the ECDSA scheme into a CVP problem and thence to an SVP problem.

- The lattice dimension we use is $n+2$ where n is the number of signatures.

- Whether n signatures with L bits of leakage per signature is enough to recover k depends on several factors.

- Clearly there is an information theoretic minimum: we need

$$n \cdot L > \log_2(q) = N.$$

- Actually, knowing the public key $[x]P$ enables attack to go beyond this minimum.

Putting It All Together

- We can use the Gaussian heuristic to see if Kannan's embedding technique or some other CVP solver is likely to produce a solution – see the exercises for an important wrinkle in this approach.

- You will implement all this in the lab and use fpylll as a tool to solve the lattice instances that arise.

- Through this programming exercise, you'll explore the performance of this kind of attack and deal with some of the subtleties that arise.

- Have fun!