Lattice Attacks in The Wild, Theory, and CTF

rkm0959

Super Guesser

June 25th

Outline

Table of Contents

Fundamentals

Definitions

- **Lattices** : Vectors in $\mathbb{Z}^n$. Linear Combinations. Coefficients in $\mathbb{Z}$.
- **SVP** : Find the shortest non-zero vector in the lattice
- **CVP** : Find the closest vector to the given one in the lattice

Using Lattices

- **Designing Cryptosystems** : SVP/CVP are quite hard. PQCrypto.
- **Breaking Cryptosystems** : RSA, HNP, Knapsack

Today's Topic : let's break things! but how to do it?

Cryptographer's Toolkits : an Introduction

LLL : Given $b_1, \cdots, b_n \in \mathbb{Z}^n$, returns vector with norm no more than

$$(4/3+\epsilon)^{(n-1)/4} \det(L)^{1/n}, \quad (4/3+\epsilon)^{(n-1)/2} \lambda_1(L)$$

for any $\epsilon > 0$ in polynomial time $O(n^6 \max \log^3 ||b_i||)$

BKZ : Time/Quality Tradeoff with blocksize β

BKZ 2.0 : Better BKZ, lots of heuristics

Babai's Algorithm : After LLL, gives $2^{n/2}$-approximation of CVP

Framework

General Framework for Lattice Attacks

Mathematical Problem $\xrightarrow{\text{Encode}}$ SVP/CVP $\xrightarrow{\text{Lattice Algorithm}}$ Solved!

Today's Focus : Context of the Attack vs Framework for the Attack

Table of Contents

Rough Idea

Framework for Lattice Attacks in the Wild

$$\text{Setup} \xrightarrow{\text{Initial Attack}} \text{Problem} \xrightarrow{\text{Encode}} \text{SVP/CVP} \xrightarrow{\text{Lattice Algorithm}} \text{CVE!}$$

- "pure math attacks" are rare in the wild (exception : ROCA)
- $\implies$ require extra attack to gather information
- proofs $\ll$ practicality, severity

The Extra Attack

It seems that usually, the extra element is **Side-Channel Attacks**.

Key Step for Lattice Attacks in the Wild

Faulty Implementation $\xrightarrow{\text{Side-Channel Attack}}$ Mathematical Problem

Side-Channel Attacks leak **bias**. Encode this into Lattices.

Examples : ROHNP (CVE-2018-0495)

Take a look at ECDSA implementation. Find a weakness.

```
def Mod(a, n):
    if 0 <= a < n:
        return a
    return a % n

# hash, secure_random : cryptographically secure
# secret key = x, message to sign = msg
z = Mod(hash(msg), n)
k = secure_random(1, n)
kinv = inverse(k, n)
r = (k * G).x
rx = Mod(r * x, n)
tot = Mod(rx + z, n)
s = Mod(kinv * tot, n)
return (r, s)
```

Examples : ROHNP (CVE-2018-0495)

Input-Dependent Behavior : The vuln is at $tot = Mod(rx + z, n)$. We know $0 \leq rx + z < 2n$, and if $rx + z < n$, no modulo operations are used.

How to reliably check this? **Flush+Reload Attack**
i.e. monitor some offsets that correspond to function computation

Examples : TPM-FAIL (CVE-2019-11090, CVE-2019-16863)

Timing Attack : If the random nonce k has more leading zero bits, (LZB) the runtime of the targeted ECDSA implementation decreases $\rightarrow$ **bias!**
Similar : CVE-2011-1945, CVE-2019-13628, CVE-2019-14317

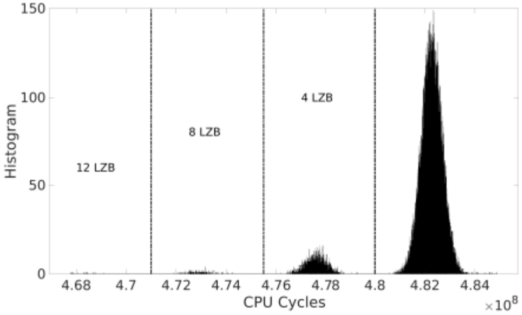

Figure 1: From TPM-FAIL Paper - Intel fTPM ECDSA (NIST-256p) Timing

Table of Contents

Rough Idea

Key Step for Lattice Attacks in Theory

Mathematical Problem $\xrightarrow{\text{Encode}}$ SVP/CVP $\xrightarrow{\text{Lattice Algorithm}}$ Solved!

- proofs have significant meaning, but heuristics are still very useful
- attacks that doesn't sound too feasible still has significant value
- two problems to research : **Encoding** and **Lattice Algorithm**

Case Study : RSA Cryptanalysis

RSA Cryptanalysis with Lattices : Usually, we use

- Howgrave-Graham Theorem, which encodes the problem into SVP
- Construct appropriate lattice, and use LLL bounds
- The challenge lies in **Lattice Construction (choosing polynomials)**

Case Study : Hidden Number Problem

HNP : Given $\{t_i\}, q$, and some MSB's of $\alpha t_i \pmod{q}$: find α.

A "natural" lattice exists for this problem $\rightarrow$ SVP/CVP instance!

Issue : solution of the SVP/CVP vs solution of the main problem

Case Study : Hidden Number Problem

Solution : prove that lattice vector in the possible output range is unique!

Possible Output Range : guaranteed by

- the length of the vector we wish was the solution
- the bounds guaranteed by the lattice algorithm

Theorem (Informal, Boneh, Venkatesan 1996)

Denote $n = \lceil \log q \rceil$, $k = \lceil \sqrt{n} \rceil + \lceil \log n \rceil$, $d = 2\lceil \sqrt{n} \rceil$. *Then,*
d *random* $\{t_i\}$*'s with* k *leaked MSB's* $\rightarrow$ *correct* α *with high probability*

Note : Similar storyline for Low-Density Knapsack (CJLOSS Algorithm)

Issue : HNP attacks work better than proved, heuristics are powerful

Kannan Embedding

Given a CVP instance with lattice B and target vector v, solve SVP on

$$\begin{pmatrix} B & 0 \\ v & t \end{pmatrix}$$

with some appropriate embedding factor $t \in \mathbb{N}$.

- In HNP, Kannan Embedding outperforms CVP approaches
- However, the attack is sensitive to the choice of t

Gaussian Heuristic

Gaussian Heuristic : Expected length of shortest vector is

$$\text{Gaussian}(L) = \sqrt{\frac{n}{2\pi e}} \cdot |\det(L)|^{1/n}$$

A good heuristic is to use L such that if our "desired" shortest vector is v,

$$\text{Gaussian}(L) : ||v||$$

is large, i.e. v is much smaller than the "usual" shortest vector.

This heuristic is suitable for HNP, and leads to many more heuristics.

Heuristic 1 : Recentering

Idea : Fix $\det(L)$ while reducing the size of $||v||$.

Consider lattice B, and we want some $v \in B$ such that

$$lb_i \leq v_i \leq ub_i, \quad \forall i$$

This is intuitively a CVP instance. **Recentering** selects target vector

$$v = \left(\frac{1}{2}(lb_1 + ub_1), \cdots, \frac{1}{2}(lb_n + ub_n)\right)$$

After this, Kannan Embedding can be used to change the problem to SVP.

Heuristic 2 : Scaling

Idea : Maximize Gaussian(L) : $||v||$ with some simple operations.

Consider lattice B, and we want some $v \in B$ such that

$$|v_i| \leq M_i, \quad \forall i$$

This is intuitively a SVP instance. **Scaling** scales each column by S_i so that

$$S_i M_i \approx C, \quad \forall i$$

for some value C. This “balances” the power of each column.

Heuristic 2 : Scaling

If we scale each column by S_i, the ratio we want to maximize is

$$\frac{\text{Gaussian}(L) \cdot (S_1 S_2 \cdots S_n)^{1/n}}{\left(S_1^2 M_1^2 + \cdots + S_n^2 M_n^2\right)^{1/2}}$$

From the AM-GM inequality, we know

$$\left(S_1^2 M_1^2 + \cdots + S_n^2 M_n^2\right)^{1/2} \geq \sqrt{n} \cdot (S_1 \cdots S_n)^{1/n} \cdot (M_1 \cdots M_n)^{1/n}$$

with equality when $S_1 M_1 = S_2 M_2 = \cdots = S_n M_n$, which gives scaling.

Guessing Bits: Improved Lattice Attacks on ECDSA

If number of leaked nonce bits are small, HNP attack is hard.

We compensate for this by **guessing**, i.e.

- Guess **nonce MSB** that were not leaked
- Guess **key MSB** that we do not know yet
- Utilize signatures with small t_i so that key LSBs do not matter
- Perform multiple LLL/BKZ algorithms in a single batch

Result : less signatures required for attacks in the wild (TPM-Fail)

Details : `https://eprint.iacr.org/2021/455.pdf`

Table of Contents

Rough Idea

It's a competition, every challenge is possible of course.

Question : How to make Lattice Attacks easier for newcomers?
My Answer 1 : Automation of **Encoding and Lattice Algorithm**
My Answer 2 : Fixed formulation of "Mathematical Problem"

Key Idea for rkm0959's Lattice Repository

System of Linear Inequalities $\xrightarrow{\text{Magical Automated Solver!}}$ Flag!

https://github.com/rkm0959/Inequality_Solving_with_CVP

Basic Idea

The problem we solve is **System of Linear Inequalities**, i.e.

$$lb_j \le \sum_{i=1}^{n} a_{ij} x_i \le ub_j, \quad \forall\, 1 \le j \le m$$

given (a_{ij}), (lb_j), (ub_j). This can be viewed as finding a vector "between"

$$lb = (lb_1, \cdots, lb_m), \quad ub = (ub_1, \cdots, ub_m)$$

in the lattice generated by the rows of $B = (a_{ij})$. View this as CVP.

The solver used CVP formulation with Recentering and Scaling.

Example : HNP

For example, HNP with n datasets $\{(t_i, MSB_i, k_i)\}$ can be viewed as

$$MSB_i \cdot 2^{k_i} \leq t_i\alpha - qz_i < (MSB_i + 1) \cdot 2^{k_i}$$
$$0 \leq \alpha < q$$

which has $n + 1$ inequalities and $n + 1$ variables $\alpha, \{z_i\}$.

Update 1 : More Utilities

Approximating Number of Solutions : If $n = m$, the number of solutions can be heuristically approximated as

$$\frac{\prod_{i=1}^{n}(ub_i - lb_i + 1)}{|\det(B)|}$$

Retrieving the Variables : If we found the lattice vector v between lb and ub, we can recover the values of x_i by solving a linear equation.

Special Case : In the case where the system is of the form

$$L \leq Ax + By \leq R, \quad S \leq x \leq E$$

there is a provable algorithm that calculates all solutions efficiently.

Update 2 : More Heuristics

Work on the constraint that $n = m$, which is common in CTFs.

Kannan Embedding : We apply Kannan Embedding and use SVP.

Algorithm Choice : We can now select between LLL and BKZ.

The Factor : Kannan Embedding Factor t is chosen to maximize

$$\text{Gaussian}(L) : ||v||$$

where v is the "expected" shortest vector of the lattice.

More work is needed on selecting the Kannan Embedding Factor.

Challenges & The Dream

Challenges

- Is this actually easier than learning lattices?
- Does this algorithm work well enough for most/all CTF challenges?
- How do we improve the heuristics used in this algorithm?
- Algorithms specific to a problem work better (CJLOSS)

My Dream

- well-made frameworks have the power to change a narrative
- be like Coppersmith's Algorithm (defund, ubuntor, mimoo)