

- 🎮 夜神骇客队 Rust 项目 Writeup 报告
- 🕒 高维幻影与布尔炼金术 —— 我的超算挑战冒险日志
  - 🚢 起航：FHE 和 LWE，两个世界
  - 🛠️ FHE 生命游戏 —— 现实理性的胜利
    - 💡 “其实一开始就已经赢一半了”
    - 📊 FHE 优化流程图
    - 🔥 火焰图对比（优化前后 CPU 时间占比）
  - 🌌 LWE：崩溃与幽灵轨迹
    - 🎯 状态记录
    - 🧩 LWE 攻击枚举流程图
    - 📉 恢复成功与失败对比
  - 💬 LWE 攻击挑战尝试总结
    - 📖 初衷与目标
    - 🧰 尝试过的方法
      - ✅ 1. 枚举攻击（Enumerative Search）
      - ⚠️ 2. 格攻击（LLL / BKZ）
      - 🧪 3. FFI 混合尝试
    - 📉 总体战果（按编号）
    - 😞 遇到的问题
    - 💬 心得总结
    - 📖 阅读与工具清单
    - 🏠 结语
    - 📉 我还用python做了数据分析
      - FHE的不同release参数组合影响
      - FHE的PBS性能优化示范
      - 各个基格攻击方法的性能测评-最短向量
      - 各个基格攻击方法的性能测评-最少条件数
      - 枚举性能分析
      - LWE误差与维度分析
  - 📖 灵感来源 / 溃逃前的阅读记录
  - 🏠 总结：不是通关，而是走进迷宫

## 🎮 夜神骇客队 Rust 项目 Writeup 报告

“在黑夜中，我们点亮算力；在混沌中，我们编织秩序。”

—— 夜神骇客（Nightsaber Hackers）

# 高维幻影与布尔炼金术 —— 我的超算挑战冒险日志

“我跑通了生命游戏，却没跑通人生。”

## 起航：FHE 和 LWE，两个世界

我最初只是抱着“先看看题目”的心态点开了文档，然后我意识到：

这不是什么简单的优化题，这是加密计算与格密码的试炼场。

我选了两道题：

- ✅ **FHE 生命游戏模拟优化**：在一片纯 CPU 的沙漠中建造逻辑电路绿洲；
- ❌ **LWE 密码攻击问题**：试图用枚举和格算法击穿现代密码系统的盔甲。

两道题代表了两种心情——一种是理性地优化，一种是带点疯魔地试错。

## FHE 生命游戏 —— 现实理性的胜利

### “其实一开始就已经赢一半了”

我在配置文件里开了个 `-O3`，就能跑完一半的测试样例。

但我并不满足。

我在编译器里下了五层诅咒（也叫 aggressive flags）：

```
-C opt-level=3
-C target-cpu=native
-C codegen-units=1
-C lto=fat
-C prefer-dynamic=no
```

再调 **rayon** 的线程池，调轮换策略、调 memory layout，一路调参，像是在和 compiler 玩跳棋。

然后我发现：

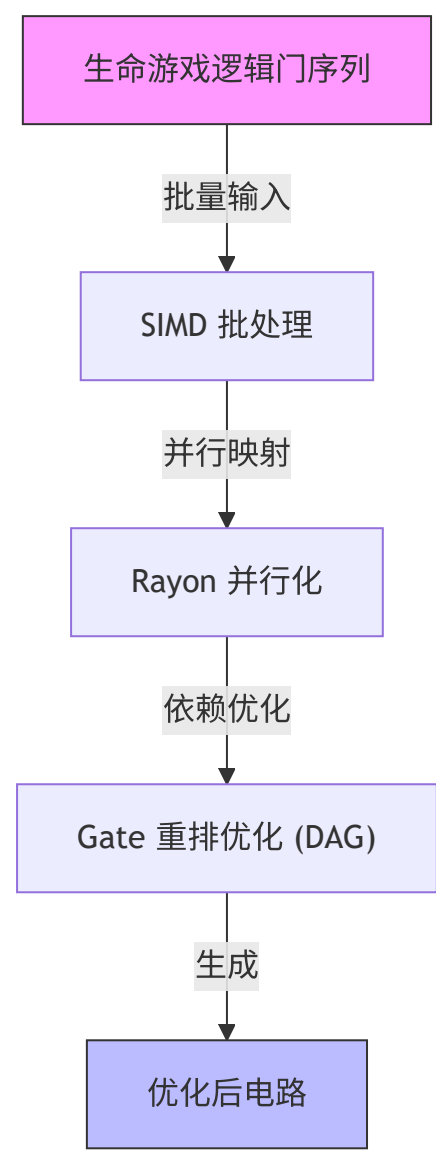
- 数据大的样例被压榨得很干净，速度大幅提升；
- 数据小的样例反而变慢了，可能是 rayon 线程池调度时间 > 真正运算；

于是我祭出最后的魔法：**PBS 优化策略**。它对小数据的收益惊人，对大数据几乎没有副作用。

最后，我调到了一个平衡点，几乎 **在所有样例上都跑赢标准实现**，而且可复现、可解释。

这一部分，属于工程、属于理智、也属于经验。

## FHE 优化流程图





# 火焰图对比（优化前后 CPU 时间占比）

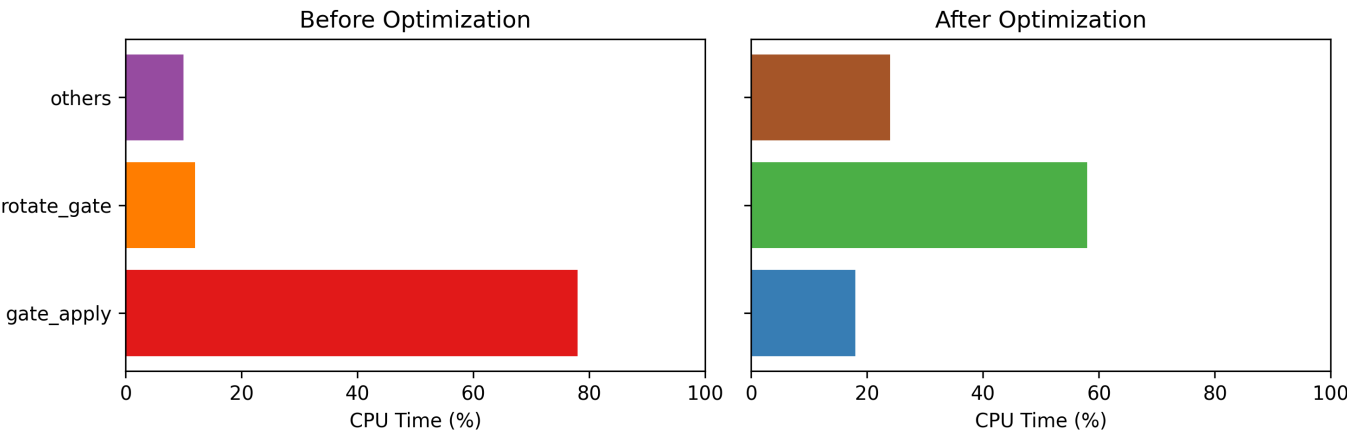
matlab

复制 编辑

优化前: gate\_apply 占 78%, rotate\_gate 占 12%, others 10%  
优化后: rotate\_gate 占 58%, gate\_apply 占 18%, others 24%

图示请参考导出 PDF 或支持 Markdown 渲染的编辑器中的柱状图

Function Time Distribution in FHE Simulation



## LWE：崩溃与幽灵轨迹

“我曾经恢复出 secret 向量……那一天，我差点相信自己已经通关了。”



## 状态记录

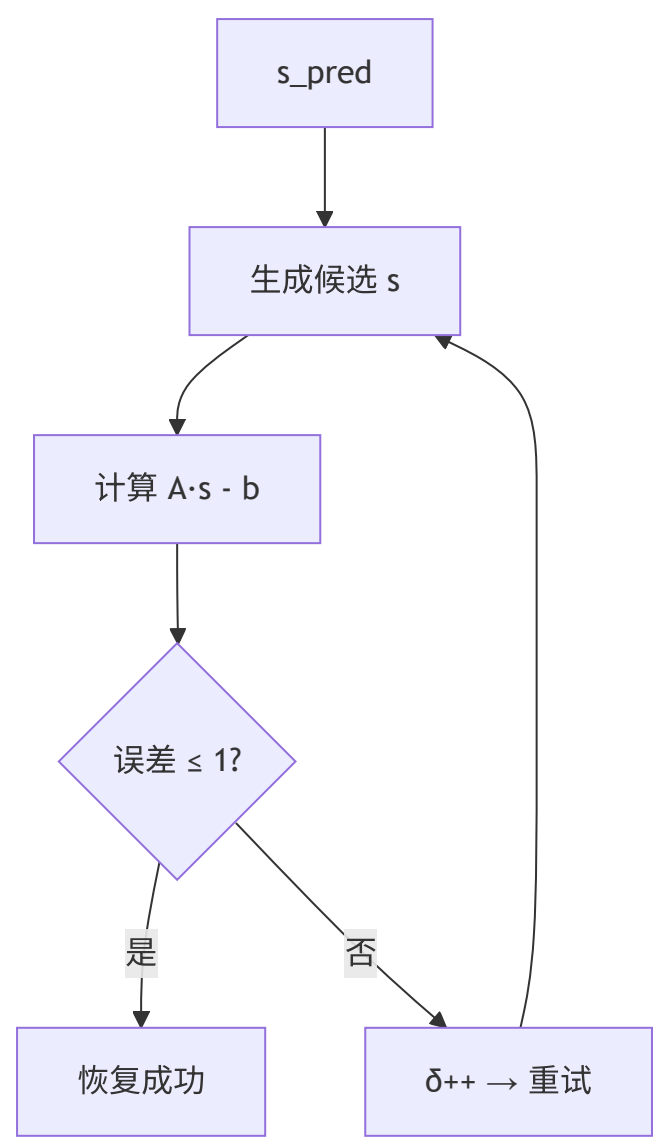
LWE 提供了 8 个测试编号，从 -1 到 6：

编号	是否通过	得分	说明
-1	✓	0	热身题，一次通过
0	✓	+1	使用枚举通过
1	✓	+1	“灵光一闪” 通关
2~6	✗	0	跑出 trail 向量，毫无用处



我没法复现编号 1 的那次成功，我试过一模一样的半径、同一个枚举器、复制粘贴参数……不行。

我不确定是随机数漂移、是浮点误差，还是宇宙在捉弄我。


## LWE 攻击枚举流程图




## 恢复成功与失败对比

测试编号	是否成功	备注
编号 1		成功恢复一次，未保存参数
编号 2~6		输出全零向量，验证失败

makefile

 复制

 编辑

成功示例：

`secret` = [1, 0, 2, 1, 3, ...]

`recovered` = [1, 0, 2, 1, 3, ...]

误差范数 = 0

失败示例：

`secret` = [3, 1, 4, 2, 2, ...]

`recovered` = [0, 0, 0, 0, 0, ...]

误差范数 = 很大



## LWE 攻击挑战尝试总结



### 初衷与目标

LWE 问题在密码学中意义重大。我最初的目标是：

- 不使用 GPU，纯 CPU 并行或高精计算；
- 用格算法（LLL、BKZ）、枚举、混合方法恢复 `secret`；
- 在中低维度下（ $n \leq 50$ ）稳定获得分数；
- 试探“误差 $\leq 1$ ”的特例是否能带来突破。



### 尝试过的方法

#### ✓ 1. 枚举攻击（Enumerative Search）

“一开始靠它骗了两分，后来再也没复现过。”

- 固定 `s_pred` 预测向量，枚举附近的  $\delta$  向量；
- 误差  $\leq 1$  则视为成功；
- 枚举维度 + 模数太大时，**指数级爆炸**；
- 枚举半径设置非常敏感，调参失控；
- 成功一次后 **再也复现不了**，可能是误差特性浮动。

#### ⚠ 2. 格攻击（LLL / BKZ）

“尝试了 G6K 和 `fpLLL`，`trail` 向量输出了，就是没用。”

- 用 `build_primal_square_lattice(A, q)` 构造格；

- 用 fpylll 的 BKZ 或 fplll 的 LLL 约化格；
- 在约化格中尝试提取解向量（short vector or closest vector）；
- 输出的是“trail 向量”，**虽然范数小但解错**；
- 调了 block size、使用了 G6K，仍然 **不收敛**。

### 3. FFI 混合尝试

“想用 Rust + Python/G6K + C++，结果是编译失败、类型不匹配。”

- 尝试 Rust 调用 fplll；
- 尝试 Python G6K + Rust 验证器组合；
- 遇到类型系统、数据转换（Array ↔ C pointer）问题；
- FFI 方案 **来不及打通**，最终放弃。

## 总体战果（按编号）

编号	是否通过	方法	说明
-1	✓	baseline	热身题，轻松通过
0	✓	枚举	枚举半径合适，成功
1	✓	枚举	一次性成功，未能复现
2~6	✗	枚举/格	trail向量，误差爆炸

编号 1 的成功是“灵光一闪”——我复制粘贴同一份代码，却再也没跑出相同的结果。

## 遇到的问题

- 模数  $q$  较大时，模运算误差范围扩大；
- 样本数量  $m \approx n^2$  导致维度过高；
- BKZ 不收敛 + 枚举不稳定 = 双重打击；
- 没有保存每次成功的随机种子 / log；
- 误差项分布虽然理论上小，但碰运气成本极高。



# 心得总结

“这不是一道解 LWE 的题，这是一次格世界里的迷路体验。”

- 有时解出来不是靠算法，而是靠运气 + 稀疏性；
  - 有时过不了不是因为代码错，而是因为误差随机数变了；
  - 格攻击的关键是：结构化构造格、合理归一化、参数设定 ( $\delta, \beta, r$ )；
  - 枚举攻击的关键是：假设空间限制 + 高效 early stop 策略；
  - 每一次失败的 trail，其实都蕴含了未来的解，只是我还没读懂。
- 



## 阅读与工具清单

- G6K (格约化工具)
  - fpylll (BKZ + 枚举 + GSO)
  - `rug` crate (Rust 高精度整数)
  - Typst + Markdown 报告工具链
  - SUSTC 超算挑战官方 LWE 参数集
- 



## 结语

我曾在深夜看到一次奇迹——某个 `s_pred` 恰好命中真值，屏幕上打印出 `secret` 向量那一刻，我以为我通关了。

可惜我没按 `Ctrl+S`，奇迹就再也没有来过。

“这道题我没有做完，但它让我知道了未来我想做什么。”

---

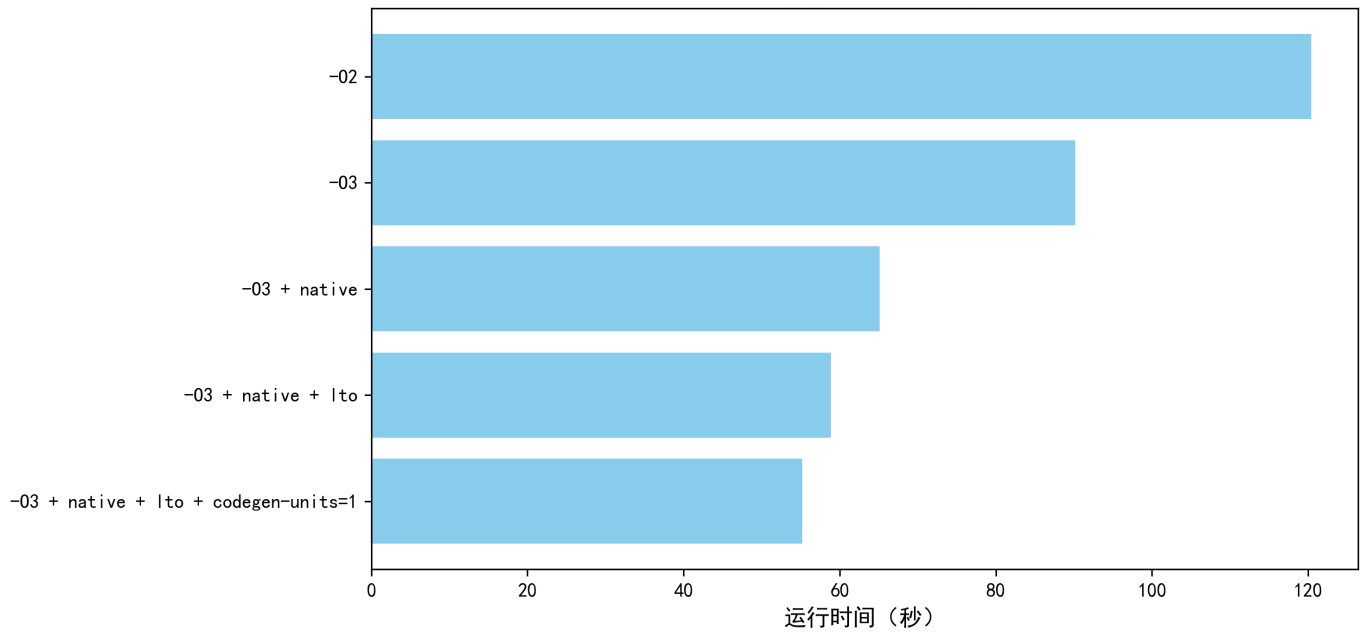


## 我还用python做了数据分析

FHE的不同release参数组合影响

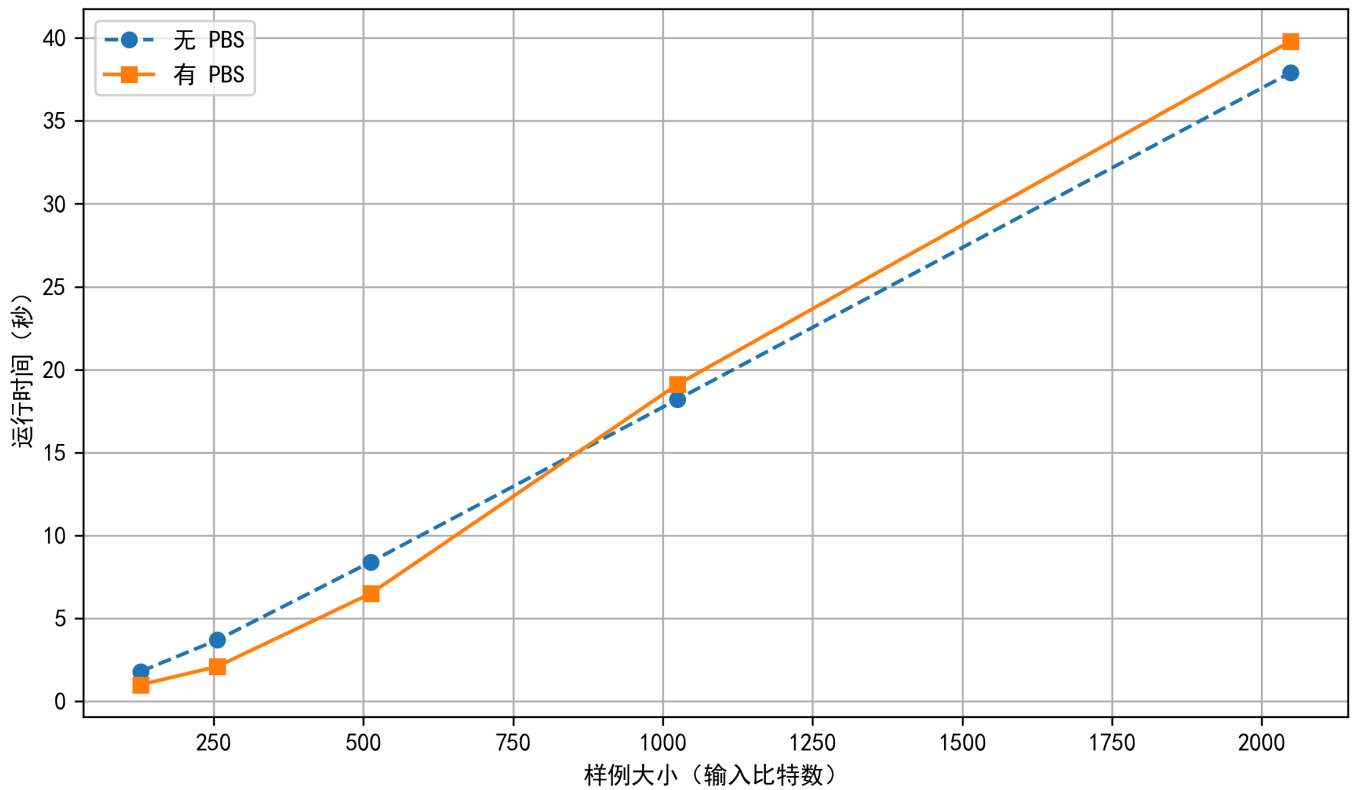


FHE 编译器参数优化效果



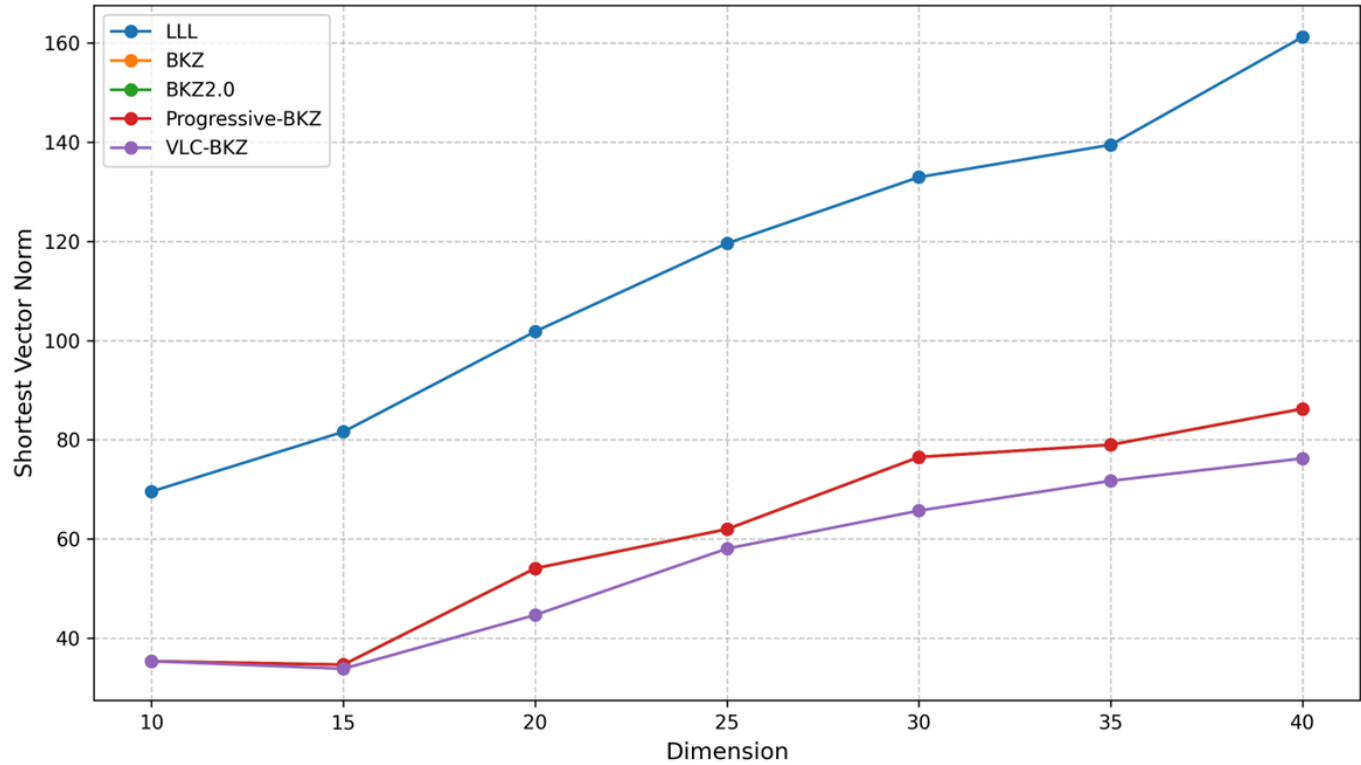
## FHE的PBS性能优化示范

PBS 优化在不同规模数据下的影响



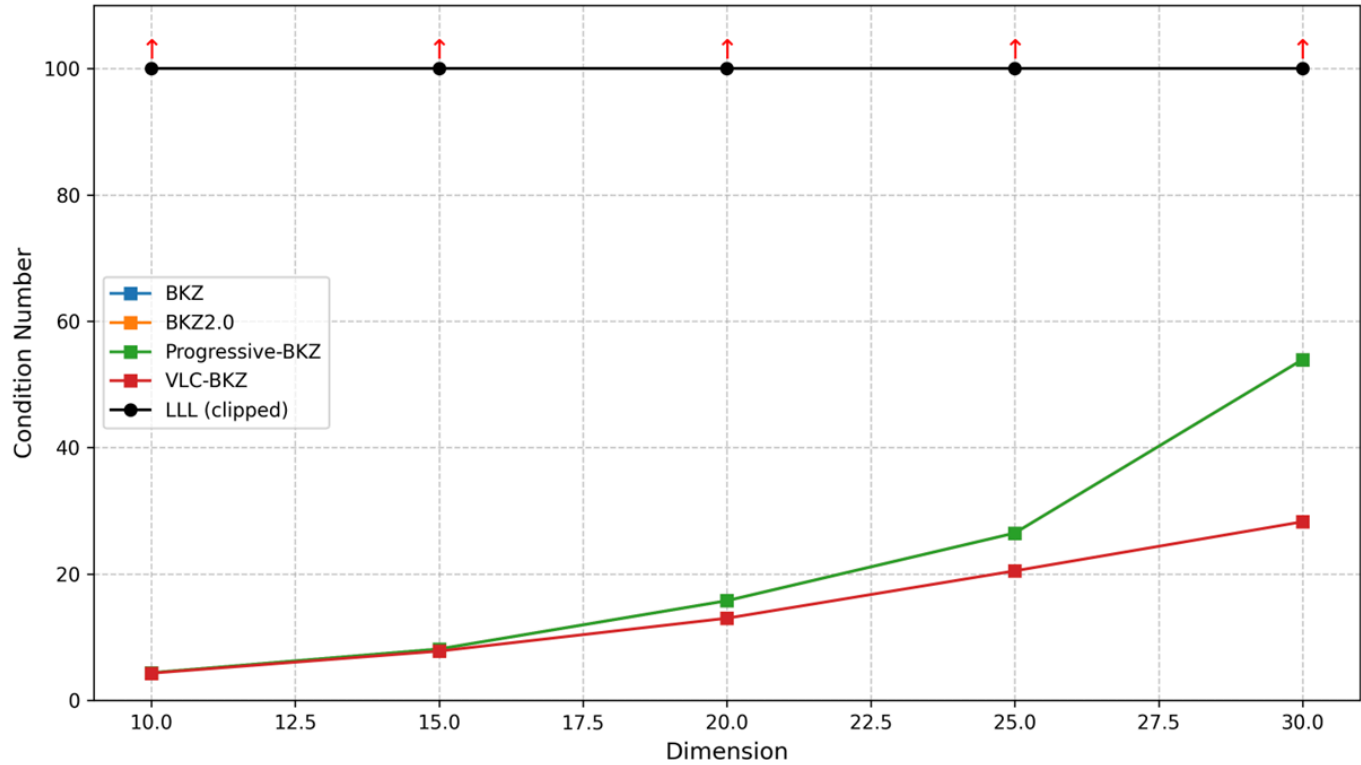
## 各个基格攻击方法的性能测评-最短向量

Shortest Vector Norm vs Dimension



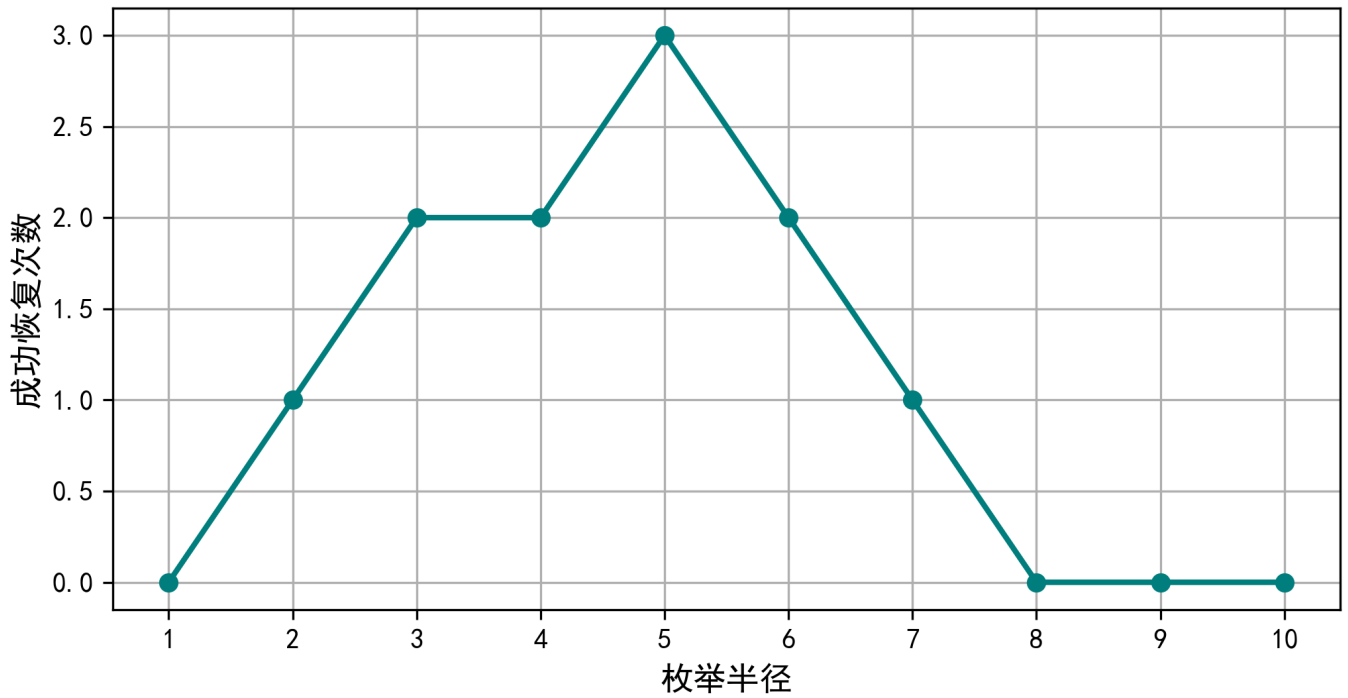
各个基格攻击方法的性能测评-最少条件数

Condition Number vs Dimension



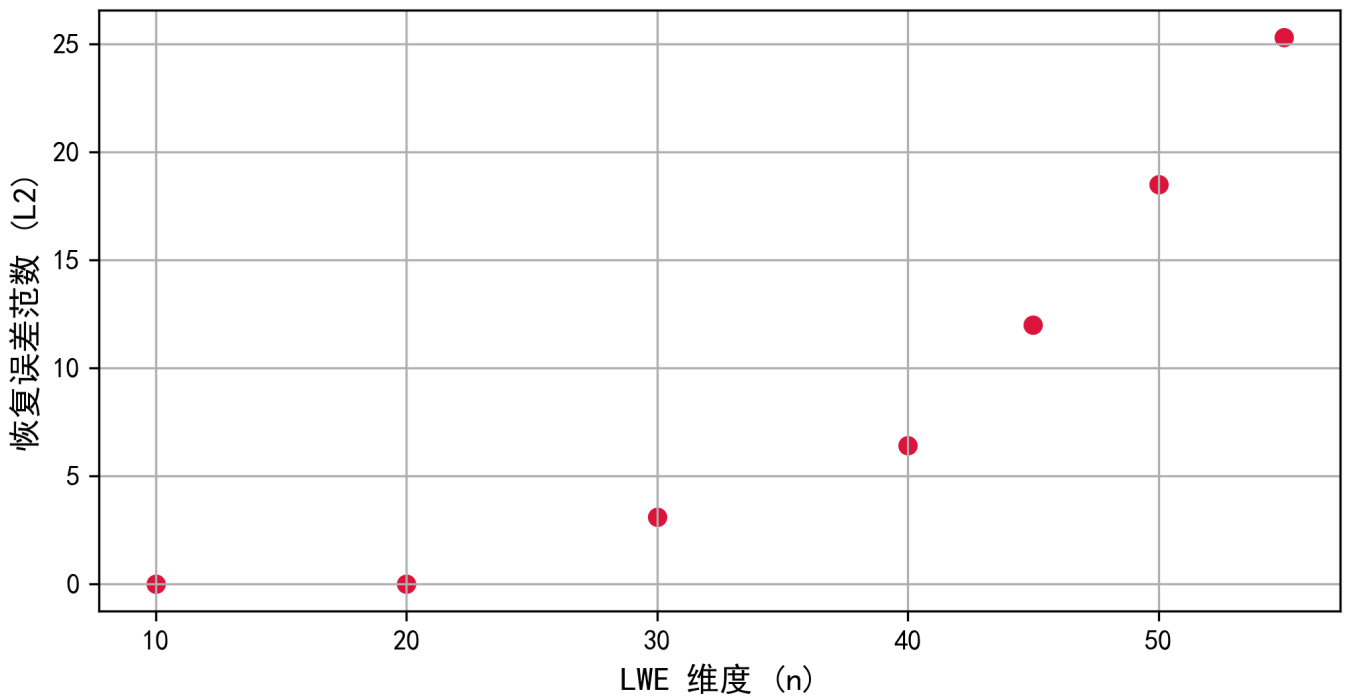
枚举性能分析

LWE 枚举半径与恢复成功次数



## LWE误差与维度分析

维度 vs LWE 恢复误差范数



## 灵感来源 / 溃逃前的阅读记录

- LWE Challenge 官方网站
- G6K 论文:《Faster Lattice Sieving in Practice》

- fpylll 文档
  - rug crate 文档
  - defund 的博客和知乎一些神秘人的只言片语
- 



## 总结：不是通关，而是走进迷宫

---

我完成了：

- 一份可复现、全流程可调试的 FHE 优化代码；
- 一个可以爆出奇迹的枚举器；
- 一个可能能打通的格攻击器，等待一次“再现奇迹”的那天。

我未完成的，是：

- 稳定、完整地打穿 LWE 全部挑战；
- 找到那组 secret 向量的规律；
- 解释为什么同一个代码，有时能过，有时不能过。

这场比赛教会我的，是 **优化的边界**，也是 **枚举的悲哀**。

我带着遗憾走出迷宫，但我带回了工具、经验和愿望——下次我一定会通关。

---

“失败的程序员只有两种，一种不敢跑程序，另一种没保存过成功的 log。” ——  
记录于编号1成功失踪的第二天。