Blockwise Reduction and Security Estimates

Phong Nguyễn

http://www.di.ens.fr/~pnguyen
SVP Algorithms

○ **Poly-time approximation algorithms.**
  ○ The LLL algorithm [LLL82].
  ○ Block generalizations by [Schnorr87, GHKN06, GamaN08, MiWa16, ALNS20].

○ **Exponential exact algorithms.**
  ○ Poly-space enumeration [Pohst81, Kannan83, ScEu94]
  ○ Exp-space sieving [AKS01, MV10].
Blockwise Algorithms
LLL is based on a local reduction in dim 2.

Blockwise algorithms find shorter vectors than LLL by using an exact SVP-subroutine in low dim k called the blocksize.

This subroutine can be done using $2^{O(k)}$ poly-time operations [AKS01,MV10,ADRS15], which is poly in $d$ if $k = \log d$. 
Mathematical Analogy

- If we show the existence of very short lattice vectors in dim $k$, can we prove the existence of very short lattice vectors in dim $d > k$?

- [Mordell1944]'s inequality generalizes Hermite's inequality:

$$\sqrt{\gamma_d} \leq \sqrt{\gamma_k^{(d-1)/(k-1)}}$$

$$\lambda_1(L) \leq \sqrt{\gamma_k^{(d-1)/(k-1)}} \text{vol}(L)^{1/d}$$
Approximation Algorithms for SVP

- Related to upper bounds on Hermite’s constant, i.e. proving the existence of short lattice vectors.
- \([LLL82]\) corresponds to \([Hermite1850]\)’s inequality.
  \[
  \lambda_1(L) \leq \sqrt{\gamma_2^{d-1}} \frac{\text{vol}(L)^{1/d}}{d} = \left(\frac{4}{3}\right)^{(d-1)/4} \frac{\text{vol}(L)^{1/d}}{d}
  \]
- Blockwise algorithms \([\text{Schnorr}87, \text{GHKN06, GN08,MW16,ALNS20}]\) are related to \([\text{Mordell1944}]\)’s inequality.
  \[
  \lambda_1(L) \leq \sqrt{\gamma k^{(d-1)/(k-1)}} \frac{\text{vol}(L)^{1/d}}{k-1}
  \]
Mordell’s Inequality (1944)

- Hermite’s inequality is the $k=2$ particular case of Mordell’s inequality:

$$\gamma_d \leq \frac{(d-1)}{(k-1)} \frac{\gamma_k}{k} \quad \text{if} \quad 2 \leq k \leq d$$

- All known proofs of Mordell’s inequality are based on duality.
Mordell’s Proof

- For Hermite’s inequality, the lattice rank was decreased by considering the quotient $L \mod b_1$.

- Duality provides another way to reduce dimensions:
  - If $L$ is a $d$-rank lattice and $v \in L^\times$ is non-zero, then $L \cap v^\top$ is a $(d-1)$-rank sublattice.
More Details


Random Lattices and Average-Case Behaviour
Average-Case Behaviour

- Experimentally [MiWa16], not many differences between blockwise algorithms, despite different theoretical bounds.

- The old BKZ algorithm [ScEu94,CN11] is widely used in lattice record computations.
BKZ Issues: Output Quality

- Theoretical worst-case bounds >> Practice.

$cd$ vs $c'd$ with $c > c'$

Same phenomenon as LLL
Predicting BKZ [CN11, BSW18]

- Predicts the approx behaviour of high-blocksize BKZ \((k \geq 50)\), using an efficient simulation algorithm: the minimum of most \(k\)-rank blocks seems to behave like random lattices.

\[
\sqrt{\frac{(d-1)}{k}} \quad \text{becomes roughly} \quad \text{GaussHeurist}(k)^\frac{(d-1)}{(k-1)}
\]
Blocks vs Random Lattices

![Graph showing average value for local block and standard deviation for local block compared to average value for random lattices.](image-url)
Security Estimates
Security Estimates

- Somewhat independent of security proofs
- Identify the best attack based on the state-of-the-art
  - Find as many attacks as possible
  - Identify the "best" one
  - Select keysizes/parameters accordingly
Selecting Keysizes

- LenstraVerheul00 suggested to:
  - Model the performances of the best algorithm known, based on record benchmarks.
  - Add a security margin by speculating on:
    - Hardware improvements: Moore’s law, etc.
    - Algorithmic improvements
A hardness assumption typically asks that no algorithm running in time \( \leq T \) can solve a random instance with probability \( \geq \varepsilon \).

A complexity analysis typically says that an algorithm runs in time \( \leq T' \) for all instances (of given size).
What is Needed

- A **lower bound** on the running time of the algorithm.
- Or more information on the **distribution** of the running time: expectation and variance.
- Typically not done in cryptanalysis.
Lattice-based submissions to NIST rely on a script to assess the security level: it does not fully reflect various uncertainties.

https://estimate-all-the-lwe-ntru-schemes.github.io/docs/

The script says the best attack runs the SVP subroutine in some blocksize:

- Estimate the cost of the SVP subroutine.
- Estimate the number of calls.
Security level = $\log_2 \#\text{operations}$

- Predicted upper bounds for BKZ-Enumeration
  - [CN11, C13, AWHT16]

- SVP Challenges Records Before 2018
  - [NgVi08] sieve = $0.415 \times \text{dim}$

- New sieving records [ADHKPS18]

- Old sieving
  - Best sieve = $0.292 \times \text{dim}$
Open problem

- Efficient algorithms to approximate SVP within a polynomial factor, possibly quantum or subexponential.