FIT5124 Advanced Topics in Security

Lecture 2: Lattice-Based Crypto. II

Ron Steinfeld
Clayton School of IT
Monash University

March 2016

Acknowledgements: Some figures sourced from Oded Regev's Lecture Notes on 'Lattices in Computer Science', Tel Aviv University, Fall 2004, and Vinod Vaikuntanathan's course on Lattices in Computer Science, MIT.

Plan for this lecture (and next)

- **How secure is lattice-based cryptography?**
  - Known cryptanalysis algorithms to break γ-SVP / SIS problem: LLL algorithm and variants.
  - Average-case hardness for SIS based on worst-case hardness of γ-SVP (only mention).
  - How to choose parameters for Ajtai's hash function for a given security level?
- **How to construct lattice-based encryption schemes? (start this week if sufficient time)**
  - Learning with Errors (LWE) Problem and Bounded Distance Decoding (BDD) problem
  - Symmetric-key encryption from LWE
  - Public-key encryption from LWE: Regev's cryptosystem (2005).

Security of Lattice-Based Cryptography

- **Q1:** How should we choose the parameters q, m, n, d of Ajtai's hash function?
- **Q2:** How hard (secure) is SIS Problem?

We attempt to answer two subquestions for Q2 and return to Q1:

- **Q2a:** How hard is it to solve γ-SVP problem for an arbitrary lattice?
- **Q2b:** How hard is it to solve γ-SVP for random q-ary lattices $L_q^{\perp}(A)$, i.e. how do we know that SIS Problem is hard on 'average'? Is there a (non-negligible) subset of 'weak' matrices A for which problem is much easier than solving γ-SVP for arbitrary lattices?

Security of Lattice-Based Cryptography

Q2a: How hard is it to solve γ-SVP problem for an arbitrary lattice?

Ans: Need to understand complexity of state of the art algorithms for these problems.

A difficult, not fully understood topic!

We briefly overview of two classical algorithms (foundation for current state of the art γ-SVP algorithm known as BKZ):

- LLL lattice reduction algorithm ($\gamma = 2^{O(n)}$, time $= n^{O(1)}$).
- Enumeration algorithms, aka Fincke-Pohst enumeration ($\gamma = 1$, time $= 2^{O(n \log n)}$) – only mention.

Optimized γ vs. time tradeoff combination of those used in BKZ (aka Schnorr's block reduction) algorithm (state of the art alg. for $\gamma = n^c$, time $\approx 2^{O(n/c)}$ – only mention.

Algorithms for γ-SVP: LLL

Recall: a given lattice L has an infinite number of bases B, but all have the same FP volume $\det L$:

- Most bases B are 'bad': long lattice vectors, far from orthogonal, FP of B is very 'skewed'
- Some bases B are 'good': short lattice vectors, close to orthogonal, FP of B is $\approx$ an n-dim. cube of side length $\approx \det L^{1/n}$.

How to transform a 'bad' basis to a better one?

Use a lattice basis reduction algorithm: Given a basis B of lattice L, outputs a 'better' basis B' for L

- Algorithm performs a sequence of unimodular operations on B
  - Add integer multiple of one column to another column
  - Swap columns
- Each op. preserves basis property,'improves' basis slightly

First efficient (poly-time) reduction algorithm: LLL (Lenstra Lenstra Lovasz, 1982)

Algorithms for γ-SVP: LLL

Idea of LLL: Make basis vectors 'approximately' orthogonal.

- GSO of a basis B tells us how 'orthogonal' the basis vectors are to each other:
  $\vec{b}_i^* = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \cdot \vec{b}_j^*$, where $\mu_{i,j} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle}{\langle \vec{b}_j^*, \vec{b}_j^* \rangle}$.

What does 'approximately' orthogonal $\vec{b}_i$'s mean?

- **Small projection component:** 'small' relative projection length $\mu_{i,j}$ of $\vec{b}_i$ along previous $\vec{b}_j^*$'s ($j < i$):
  - LLL property 1: $|\mu_{i,j}| \leq 1/2$ for all $i = 1, \ldots, n$ and $j < i$.
- **Large orthogonal component:** 'large' remaining component $\|\vec{b}_{i+1}^*\|$ of $\vec{b}_{i+1}$ after removing components along $\vec{b}_j^*$'s ($j < i + 1$):
  - LLL property 2: $\|\vec{b}_{i+1}^* + \mu_{i+1,i}\vec{b}_i^*\|^2 \geq \delta \cdot \|\vec{b}_i^*\|^2$ for all $i = 1, \ldots, n-1$ for some constant δ ($1/4 \leq \delta < 1$).

Goal of LLL: Perform elementary unimodular operations until both properties are satisfied.

Algorithms for γ-SVP: LLL

Definition

A basis B for lattice L is δ-LLL reduced if both LLL properties 1 and 2 are satisfied.

LLL Algorithm. Given n-dim. input basis B, do:

- Start Step: Compute GSO B^* for B.
- Length Reduction Step: (comment: after this step, LLL property 1 will be satisfied)
  - for $i = 2$ to n do
    - for $j = i - 1$ to 1 do
    - Update $\vec{b}_i \leftarrow \vec{b}_i - c_{i,j}\vec{b}_j$, where $c_{i,j} = \left\lceil \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle}{\langle \vec{b}_j^*, \vec{b}_j^* \rangle} \right\rfloor$.
- Swap Step: (comment: after this step, LLL property 2 will be satisfied by $\vec{b}_i$, $\vec{b}_{i+1}$)
  - If there is an i such that LLL property 2 is not satisfied (i.e. $\|\vec{b}_{i+1}^* + \mu_{i+1,i}\vec{b}_i^*\| < \delta \cdot \|\vec{b}_i^*\|$), then:
    - Swap $\vec{b}_i$ and $\vec{b}_{i+1}$
    - Go back to Start Step.
  - Else, Return δ-LLL reduced basis B.

Why does LLL work - Property 1?

- After length red. Step, LLL property 1 ($|\mu_{i,j}| \leq 1/2$) is satisfied:
  - Throughout length red., GSO vectors $\vec{b}_i^*$'s do not change!
    - Adding a multiple of $\vec{b}_j$ for $j < i$ to $\vec{b}_i$ only changes the projection of $\vec{b}_i$ along $\vec{b}_j$, not the orthogonal component $\vec{b}_i^*$.
  - Recall (first lecture): $\vec{b}_i$'s coordinate matrix along the rotated coordinate system of normalized GSO vectors $\vec{b}_i^* / \|\vec{b}_i^*\|$:

$$\begin{bmatrix} \|\vec{b}_1^*\| & \|\vec{b}_1^*\| \cdot \mu_{2,1} & \cdots & \|\vec{b}_1^*\| \cdot \mu_{n,1} \\ 0 & \|\vec{b}_2^*\| & \cdots & \|\vec{b}_2^*\| \cdot \mu_{n,2} \\ 0 & 0 & \cdots & \|\vec{b}_3^*\| \cdot \mu_{n,3} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \|\vec{b}_n^*\| \end{bmatrix}$$

    - jth iteration of inner for loop: subtract integer multiple $c_{i,j} = \lceil \mu_{i,j} \rfloor$ of jth column above ($\vec{b}_j$) from ith column $\vec{b}_i$ – jth entry of ith column changes from $\|\vec{b}_i^*\| \cdot \mu_{i,j}$ to $\|\vec{b}_i^*\| \cdot (\mu_{i,j} - \lceil \mu_{i,j} \rfloor)$. So: $\mu_{i,j} \to \mu'_{i,j} = \mu_{i,j} - \lceil \mu_{i,j} \rfloor$ so $|\mu'_{i,j}| \leq 1/2$.

Algorithms for γ-SVP: LLL

Why does LLL work - Property 2? After swap step, LLL property 2 ($\|\vec{b}^*_{i+1} + \mu_{i+1,i}\vec{b}^*_i\| \geq \delta \cdot \|\vec{b}^*_i\|$) is satisfied:

- Overall effect of swap step, swapping $\vec{b}_i$ and $\vec{b}_{i+1}$:
  - $\vec{b}^*_j$ (and $\mu_{i,j}$) for $j < i$ stay the same: $\vec{b}^{*new}_j = \vec{b}^*_j$ for $j < i$.
  - $\vec{b}^*_i$ and $\vec{b}^*_{i+1} + \mu_{i+i,i}\vec{b}^*_i$ swap so property 2 at i is satisfied after swap:
    - $\vec{b}^{*new}_i = \vec{b}^*_{i+1} + \mu_{i+i,i}\vec{b}^*_i$.
    - $\vec{b}^{*new}_{i+1} + \mu^{new}_{i+i,i}\vec{b}^{*new}_i = \vec{b}^*_i$

Algorithms for γ-SVP: LLL

Why does LLL work - Run time?

Swap step may invalidate property 1 while length reduction step may invalidate property 2, but...

It can be shown that this cannot continue for very long - eventually both properties 1 and 2 are satisfied and algorithm terminates!

Theorem

The number of (length reduce, swap) iterations of LLL on input basis B before termination is at most

$$n^2 \cdot \log(\max_i \|\vec{b}_i\|)/\log(1/\sqrt{\delta}).$$

For any constant $1/4 < \delta < 1$, this is polynomial in bit length of the algorithm input. Moreover, the run-time for each iteration is also polynomial in the input bit length. Overall, run-time is polynomial in input length — LLL is efficient!.

Algorithms for γ-SVP: LLL

How can we use LLL to solve γ-SVP?

Intuitively, since LLL outputs an 'approximately orthogonal' basis for L, the basis vectors should be relatively short lattice vectors. A practical approach to solve γ-SVP for $L(B)$:

- Run LLL on B and get a δ-LLL reduced LLL basis B' for L.
- Output the shortest vector among the n basis vectors in B'.

What approx. factor γ does this achieve? Not easy to predict theoretically!

But LLL properties of B' allow us to prove an upper bound on big γ can be.

Algorithms for γ-SVP: LLL

Theoretical Upper bound on LLL Approx. factor γ. If B is δ-LLL reduced basis for L, LLL property 2 is:

$$\|\vec{b}^*_{i+1} + \mu_{i+1,i}\vec{b}^*_i\|^2 \geq \delta \cdot \|\vec{b}^*_i\|^2$$

By Pythagoras, LHS above is just $\|\vec{b}^*_{i+1}\|^2 + \mu^2_{i+1,i}\|\vec{b}^*_i\|^2$, so we can rearrange to get:

$$\|\vec{b}^*_{i+1}\|^2 \geq (\delta - \mu^2_{i+1,i}) \cdot \|\vec{b}^*_i\|^2.$$

By LLL property 1, $\mu^2_{i+1,i} \leq 1/4$ so we get the successive ratio bound:

$$\frac{\|\vec{b}^*_{i+1}\|^2}{\|\vec{b}^*_i\|^2} \geq (\delta - 1/4), \text{ for all } i \geq 2.$$

Since the ratio of norms of all pairs of successive $\vec{b}^*_i$'s is at least $\delta - 1/4$, it immediately implies

$$\frac{\|\vec{b}^*_n\|^2}{\|\vec{b}^*_1\|^2} \geq (\delta - 1/4)^{n-1},$$

or $\|\vec{b}_1\| = \|\vec{b}^*_1\| \leq (1/(\delta - 1/4))^{(n-1)/2} \cdot \|\vec{b}^*_n\|$. But it can be shown that $\|\vec{b}^*_n\| \leq \lambda_1(L)$, so $\|\vec{b}_1\| \leq (1/(\delta - 1/4))^{(n-1)/2} \cdot \lambda_1(L)$. Conclusion (take $\delta = 3/4$):

Theorem

The LLL algorithm solves (in polynomial time) γ-SVP for n-dim. lattices, with $\gamma \leq 2^{(n-1)/2}$. Can also be shown that Hermite Factor $\gamma_{HF} \stackrel{\text{def}}{=} \frac{\|\vec{b}_1\|}{\det(L)^{1/n}} \leq 2^{(n-1)/4}$.

Algorithms for γ-SVP: LLL

LLL Approx. factor γ in practice. For 'random' lattices, LLL experimentally performs much better than the theoretical upper bound $\gamma \leq 2^{(n-1)/2}$.

Experiments (see, e.g., [NS06]) show that for random lattices, LLL reduced bases tend to have, on average,

$$\frac{\|\vec{b}^*_{i+1}\|^2}{\|\vec{b}^*_i\|^2} \approx 1.04, \text{ for all } i \geq 2.$$

Consequently, for random lattices, LLL can (experimentally) solve γ-SVP for $\gamma \approx 1.04^{n-1}$. (and Hermite Factor $\gamma_{HF} \approx 1.02^{n-1}$).

Algorithms for γ-SVP: Enumeration

How to compute the shortest vector ($\gamma = 1$). If we really want the shortest vector, can always use a brute force search approach – enumeration algorithms.

- Enumerate all lattice vectors in a volume that is guaranteed to contain the shortest vector.
- Will not go into details here.

Drawback: enumeration run-time is (at least) exponential in dimension n!

Current state of the art enumeration algorithms (aka Fincke-Phost / Kannan) take time $2^{O(n \log n)}$.

Remark: Other algorithms (sieve algorithms – Ajtai et al 2001, Voronoi algorithms - Micciancio et al 2010, Gaussian sampling algorithms – Regev et al 2015) exist that trade off exponential memory $2^{O(n)}$ for $2^{O(n)}$ time.

- Large memory tends to make these algorithms less practical (but still being improved)...

Algorithms for γ-SVP: BKZ

Trading off larger time for smaller γ. In late 1980s, Schnorr introduced a hierarchy of generalizations of LLL, called Block Korkhine Zolotarev (BKZ) algorithm Trades off larger run-time for smaller approx. factor γ – currently state of the art for attacking lattice-based crypto:

- Combines ideas of LLL and enumeration algorithms.
- **Idea:** introduce a 'block size' parameter $k \in \{2, 3, ..., n\}$ into LLL: generalize the 2×2 GSO submatrix blocks in LLL property 2
- Gradual 'interpolation' between the extremes of LLL ($k = 2$, $\gamma = 2^{O(n)}$, $T = n^c$) and enumeration ($k = n$, $\gamma = 1$, $T = 2^{O(n \log n)}$).
- For general block size k, variants of BKZ [HPS'11] provably achieve $\gamma(k) \leq k^{(n-1)/(k-1)}$ with run-time $n^c \cdot k^{O(k)}$.

Complexity of γ-SVP: Asymptotic Summary

Summary: State of the art (BKZ) asymptotic γ-time tradeoff. For future reference, we have the following (approximate) asymptotic relations:

- For security against attacks running time $T = 2^{\lambda}$ – security parameter λ, need
$$2^{\lambda} = 2^{O(k \log k)}, \text{ so } \lambda = O(k \log k).$$
- At this run-time, achieve $\gamma_{HF} = \delta(k)^n$ with $\delta(k) = k^{1/(k-1)} \approx k^{1/k}$, so
$$\log \gamma_{HF} = (n/k) \log k, \text{ so } \log \gamma_{HF} = \Omega(\frac{n \log^2 \lambda}{\lambda}).$$

Overall, get asymptotic lattice 'rule of thumb' for γ-SVP (using BKZ):

$$n = \Omega(\frac{\lambda}{\log^2 \lambda} \cdot \log \gamma_{HF}) \approx \lambda \cdot \log \gamma_{HF}.$$

Remark: Need lattice dim. n proportional to product of bit-security level λ and log. approx. factor.

- $\log \gamma_{HF}$ factor is a reason behind relatively long keys in lattice-based cryptosystems...

Complexity of γ-SVP: Numerical Summary

Numerical estimates of optimized BKZ time versus γ. Chen and Nguyen [CN11] gave numerical estimates for Hermite Factor and time for 'random' lattices versus block size for optimized (state of the art) BKZ variants:

Table 2. Approximate required blocksize for high-dimensional BKZ, as predicted by the simulation

Target Hermite Factor	1.01^n	1.009^n	1.008^n	1.007^n	1.006^n	1.005^n
Approximate Blocksize	85	106	133	168	216	286

Table 3. Upper bound on the cost of the enumeration subroutine, using extreme pruning with aborted-BKZ preprocessing. Cost is given as $\log_2$(number of nodes).

Blocksize	100	110	120	130	140	150	160	170	180	190	200	250
BKZ-75-20%	41.4	47.1	53.1	59.8	66.8	75.2	84.7	94.7	105.8	117.6	129.4	204.1
Simulation of BKZ-90/100/110/120	40.8	45.3	50.3	56.3	63.3	69.4	79.9	89.1	99.1	103.3	111.1	175.2

Can be used to estimate concrete numerical parameters for cryptosystems!

Parameters for Ajtai's hash Function: Hardness of SIS

Recall: Ajtai's hash function collision-resistance security (provably) depends on hardness of SIS problem: finding vectors of length $\leq \beta = 2d\sqrt{m}$ in SIS lattice $L_q(A)$ (dimension m, $\det L_q(A) = q^n$ – see tute).

How to choose parameters q, n, m, d for given security parameter λ based on hardness of SIS?

To get security level $\approx 2^{\lambda}$ (enum. cost) against BKZ attacks, possible approach (see [MR08] survey):

- Assume attacker runs BKZ with block length k such that enumeration cost is $\approx 2^{\lambda}$ (e.g. use [CN11] tables).
- Find corresponding BKZ Hermite factor $\gamma_{HF} = \delta^m$ (e.g. use [CN11] tables).
- Attacker can compute a non-zero vector $\vec{v}$ in SIS lattice $L_q(A)$ of norm $\leq \ell = \min(q, \delta^m \cdot \det(L_q(A))^{1/m})$. Breaks SIS_{β} if $\min(q, \delta^m \cdot \det(L_q(A))^{1/m}) \leq \beta$.

Parameters for Ajtai's hash Function: Hardness of SIS

- Attack optimization ([MR08]): Attacker uses only a subset of $m' \leq m$ of columns of A, where m' is chosen to an optimal value m^* minimizing $\ell(m') = \min(q, \delta^{m'} \cdot \det(L_q(A))^{1/m'})$. Turns out that $m^* = \sqrt{\frac{n \log q}{\log \delta}}$ and $\ell(m^*) = \min(q, 2^{2\sqrt{n \log q \log \delta}})$.

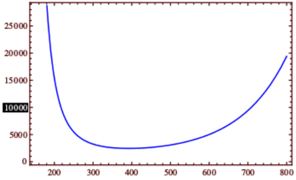

Figure 2: Estimated length of vector found with $\delta = 1.01$, $q = 4416857$, and $n = 100$ as a function of m.

- For SIS_β hardness, choose hash parameters such that $\ell(m^*) > \beta^* = 2d\sqrt{m*}$, so:

$$q \geq \beta* = 2d\sqrt{m*} \text{ and } n \geq \frac{\log^2(\beta^*)}{4 \log q \log(\delta)}.$$

Ajtai's hardness proof for SIS

Why do we think that SIS is 'hard on average' (no weak instances occur with non-negligible probability)?
Ajtai's average-case to worst-case connection Theorem (1996, improved by Gentry et al [GPV08]).

Theorem

If there is an algorithm A that solves $SIS_{q(n),m(n),\beta(n)}$ in poly-time, for some **non-negligible fraction** *of input matrices $G \in \mathbb{Z}_q^{mn \times n}$,*

Then there is an algorithm B that solves $\gamma(n)$-SIVP in polynomial time for **all** *input lattices L of dimension n with:*

$$\gamma = O(\beta\sqrt{n}), q(n) = \omega(\gamma\sqrt{\log n}).$$

- γ-SIVP is a variant of γ-SVP that asks for a γ approximation to the n linearly independent shortest lattice vectors.
- We won't study this proof, but it gives us a theoretical foundation for security of SIS.

References referred to in the Slides

NS'06 P.Q. Nguyen and D. Stehlé, LLL on the Average, In Proceedings of ANTS 2006.

GN'08 N. Gama and P.Q. Nguyen, Predicting Lattice Reduction, In Proceedings of EUROCRYPT 2008.

CN'11 Y. Chen and P.Q. Nguyen, BKZ 2.0: Better Lattice Security Estimates, In Proceedings of ASIACRYPT 2011.

MR'08 D. Micciancio and O. Regev. Lattice-Based Cryptography. Book Chapter in Post Quantum Cryptography, D.J. Bernstein; J. Buchmann; E. Dahmen (eds.), February 2009.

GPV'08 C. Gentry and C. Peikert and V. Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. In Proceedings of STOC 2008.