Pseudorandom Generator Based on Hard Lattice Problem

Cheng Kuan[*]

Abstract

This paper studies how to construct a pseudorandom generator using hard lattice problems.

We use a variation of the classical hard problem *Inhomogeneous Small Integer Solution* ISIS of lattice, say *Inhomogeneous Subset Sum Solution* ISSS. ISSS itself is a hash function. Proving the preimage sizes ISSS hash function images are almost the same, we construct a pseudorandom generator using the method in [GKL93]. Also, we construct a pseudoentropy generator using the method in [HILL99]. Most theoretical PRG constructions are not feasible in fact as they require rather long random bits as seeds. Our PRG construction only requires seed length to be $O(n^2 \log_2 n)$ which is feasible practically.

1 Introduction

Pseudorandom generator (PRG) is an important cryptographic primitive. It could stretch longer random bits from the random seed, using polynomial time. The generated bit string and the uniform distributed bit string are computationally indistinguishable.

Major study for theoretical PRG starts from 1980s. Blum and Micali [BM82] constructed the first theoretical constructor using *Discrete Logarithm*, giving the original definition of PRG. Yao [Yao82a] gave the currently used definition of PRG and proved that bit-unpredictable implies pseudorandomness. This definition is equivalent to [BM82]. Both of their construction method used one-way function. This became the common way to construct PRG for researchers after them. In this paper, we only study PRG constructed by one-way function though a long-term existed open problem is whether PRG could be constructed not based on one-way function.

Yao's method in [Yao82a] proves that PRG could be constructed from any one-way permutation. However, one-way permutation is merely existed. Most one-way function we know are hash function whose output length is shorter than input length. Levin [Levin87] weakened the requirement for one-way function, proving one-way on iteration is sufficient for PRG construction. Goldreich at al. [GKL93] provided the method that we will used in this paper. It requires each one-way function image has roughly the same number of preimage. Finally in [HILL99], it is proved that every one-way function could be used to construct a PRG. However, this method requires seed length to be $O(n^8)$ which is impractical as we require n to be at least 64. However this construction showed a new concept in information science, the pseudoentropy. Our one-way function could also be made to be a pseudoentropy generator.

Lattice problem is studied in the view of cryptography since 1996 when Ajtai [Ajtai96] proved the average-case hardness of lattice problem *Small Integer Solution* (SIS). The most recent research showed that SIS and its variation ISIS could be reduced from GapSVP [GPV08]. It is a worst-case to average-case reduction which means by random sampling we could get a hard instance of SIS or ISIS in polynomial time. We will construct our one-way function based on a variation of ISIS, say ISSS. We prove that the this one-way function meet the requirements of [GKL93] by setting proper parameters.

[*]Computer Science Department of Tsinghua University, Haidian District, Beijing, China. Email: chengk11@mails.tsinghua.edu.cn.

2 Preliminaries

2.1 Basic Notations

The sets of real numbers and integers are denoted by $\mathbb{R}$ and $\mathbb{Z}$ respectively. Vectors are represented as lower-case letters, e.g. $\mathbf{x}$. Matrix and basis are denoted by upper-case letters, e.g. $\mathbf{B}$. $\mathcal{L}(\mathbf{B})$ denote the lattice generated by basis $\mathbf{B}$. For a vector $\mathbf{x}$, the ith coordinate is denoted by x_i. The inner product between $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ is $\mathbf{x} \cdot \mathbf{y} = \langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i$.

The l_p norm of $\mathbf{x}$ is $\|\mathbf{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$ for any $p \in [1, \infty)$ and the l_∞ norm is $\|\mathbf{x}\|_\infty = \max_{1 \le i \le n} |x_i|$. We omit the subscript when $p = 2$.

The operation $\odot$ stands for matrix multiplication on GF[2]. For $x, y \in \{0,1\}^l, x \odot y = \sum_{i=1}^{l} x_i y_i \mod 2$.

For function f, we denote $f^{(0)}(x) = x$, $f^{(i)}(x) = f^{(i-1)}(x)$. $\{f^{(i)}(x), i = 0, \dots, l\}$ is a set of iterative values for iterative function with input x.

2.2 Definitions

Lattice is a $\mathbb{Z}$-module. Its dimension n could be regarded as a security parameter.

Definition 1 (Lattice). *Given n linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n \in \mathbb{R}^m$, the lattice generated by them is defined as*

$$\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i | x_i \in \mathbb{Z}\}.$$

$\mathbf{b}_1, \dots, \mathbf{b}_n$ is a lattice basis. Equivalently, if we define $\mathbf{B}$ as the $m \times n$ matrix whose columns are $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n$, then the lattice generated by $\mathbf{B}$ is

$$\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n) = \{\mathbf{B}\mathbf{x} | \mathbf{x} \in \mathbb{Z}^n\}.$$

In the following passage, without loss of generality, assume that the input basis $\mathbf{B}$ is full rank and is an integer matrix.

Definition 2 (Shortest Vector Problem (SVP)). *Given a basis $\mathbf{B} \in \mathbb{R}^{m \times n}$, find a nonzero lattice vector $\mathbf{B}\mathbf{x}$(with $\mathbf{x} \in \mathbb{Z}^n \setminus \{0\}$) such that*

$$\|\mathbf{B}\mathbf{x}\| \le \|\mathbf{B}\mathbf{y}\|$$

for any other $\mathbf{y} \in \mathbb{Z}^n \setminus \{0\}$.

Always, researchers use λ_i (also $\lambda_i(\mathbf{B})$ with respect to basis $\mathbf{B}$) to denote the ith shortest vector in a lattice. λ_1 is the shortest vector. We also use Λ to denote a lattice.

GapSVP is defined as the following.

Definition 3 (GapSVP$_\gamma$). *The input consists of $B \in \mathbb{Z}^{m \times n}$ and $r \in \mathbb{Q}$.*

- *In YES instances, $\lambda_1(\mathcal{L}(B)) \le r$.*
- *In NO instances, $\lambda_1(\mathcal{L}(B)) > \gamma \cdot r$.*

Definition 4 (Small Integer Solution (SIS$_{q,m,\beta}$)). *Given an integer q, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and a real β, find a nonzero integer vector $\mathbf{e} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{e} = \mathbf{0} \pmod q$ and $\|\mathbf{e}\| \le \beta$.*

Definition 5 (Inhomogeneous Small Integer Solution (ISIS$_{q,m,\beta}$)). *Given an integer q, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a vector $\mathbf{b} \in \mathbb{Z}_q^n$ and a real β, find a nonzero integer vector $\mathbf{e} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{e} = \mathbf{b} \pmod q$ and $\|\mathbf{e}\| \le \beta$.*

Based on ISIS, we give our definition for its variation ISSS.

Definition 6 (Inhomogeneous Subset-Sum Solution ($\mathrm{ISSS}_{q,m}$))**.** *Given an integer q, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a vector $\mathbf{b} \in \mathbb{Z}_q^n$, find a nonzero integer vector $\mathbf{e} \in \{0,1\}^m$ such that $\mathbf{Ae} = \mathbf{b}$ (mod q).*

3 Lattice One-Way Function and Its Property

3.1 The One-way Function

According to the definition of ISSS, we focus on the following hash function to construct our pseudorandom generator.

$$f_{\mathbf{A}}(x) = \mathbf{A}x, x \in \{0,1\}^m.$$

$\mathbf{A}$ is constructed by picking m random elements uniformly from a finite abelian group G. The reader could regard G as $\mathbb{Z}_q^n$.

First we will prove a special property of this one-way function.

Lemma 1 is proved according to on Claim 5.3 of [Regev05].

Lemma 1. *Let G be some finite abelian group and let l be some integer. For any l elements $g_1, \dots, g_l \in G$ consider the preimage size $|f^{-1}(h)|$ of the following function,*

$$h = f(\mathbf{b}) = \sum_{i=1}^{l} b_i g_i, \mathbf{b} \in \{0,1\}^l.$$

With probability at least $1 - \sqrt[4]{|G|/2^l}$, $\max_h |(|f^{-1}(h)| - \frac{2^l}{|G|})| \leq 2^{\frac{3}{4}l} \cdot \sqrt[4]{|G|}$.

Proof. Assume $\mathbf{g} = (g_1, \dots, g_l)$ which are l elements from G. $P_{\mathbf{g}}$ be the distribution of the sum of a random subsets of $g_1, \dots, g_l$, i.e.,

$$P_{\mathbf{g}}(h) = \frac{1}{2^l} |\{\mathbf{b} \in \{0,1\}^l | \sum_{i=1}^{l} b_i g_i = h\}|$$

According to Claim 5.3 in [Regev05], with probability at least $1 - \sqrt[4]{|G|/2^l}$,

$$\sum_h |P_{\mathbf{g}}(h) - 1/|G|| \leq \sqrt[4]{\frac{|G|}{2^l}}.$$

With both sides multiplied by 2^l and $|f^{-1}(h)| = P_{\mathbf{g}}(h) \times 2^l$, we have the following.

$$\sum_h |(|f^{-1}(h)| - \frac{2^l}{|G|})| \leq 2^{\frac{3}{4}l} \cdot \sqrt[4]{|G|}$$

As a result,

$$\max_h |(|f^{-1}(h)| - \frac{2^l}{|G|})| \leq 2^{\frac{3}{4}l} \cdot \sqrt[4]{|G|}.$$

□

Lemma 2. *G is a abelian group and l be an integer. For any l elements $g_1, \dots, g_l \in G$, consider the preimage size $|f^{-1}(h)|$ of the following function,*

$$h = f(\mathbf{b}) = \sum_{i=1}^{l} b_i g_i, \mathbf{b} \in \{0,1\}^l.$$

Suppose the decimal part of $\log \frac{2^l}{|G|}$ *is* d. *That is* $\log \frac{2^l}{|G|} = [\log \frac{2^l}{|G|}] + d$.

If G *meets the following restriction,* $0 < d + \log(1 - \frac{|G|^{\frac{3}{4}}}{2^{\frac{1}{4}l}}) < d + \log(1 + \frac{|G|^{\frac{3}{4}}}{2^{\frac{1}{4}l}}) < 1$,

$$\forall h \in G, \lceil \log |f^{-1}(h)| \rceil = \lceil \log \frac{2^l}{|G|} \rceil,$$

also

$$\forall h \in G, \lfloor \log |f^{-1}(h)| \rfloor = \lfloor \log \frac{2^l}{|G|} \rfloor.$$

Proof. According to the last lemma, we know that,

$$\max_h |(|f^{-1}(h)| - \frac{2^l}{|G|})| \le 2^{\frac{3}{4}l} \cdot \sqrt[4]{|G|}.$$

$$\forall h \in |G|, \frac{2^l}{|G|} - 2^{\frac{3}{4}l} \cdot \sqrt[4]{|G|} \le |f^{-1}(h)| \le \frac{2^l}{|G|} + 2^{\frac{3}{4}l} \cdot \sqrt[4]{|G|}$$

Compute $\log |f^{-1}(h)|$.

$$\forall h \in |G|, \log \frac{2^l}{|G|} \cdot (1 - \frac{|G|^{\frac{3}{4}}}{2^{\frac{1}{4}l}}) \le \log |f^{-1}(h)| \le \log \frac{2^l}{|G|} \cdot (1 + \frac{|G|^{\frac{3}{4}}}{2^{\frac{1}{4}l}})$$

$$[\log \frac{2^l}{|G|}] + d + \log(1 - \frac{|G|^{\frac{3}{4}}}{2^{\frac{1}{4}l}}) \le \log |f^{-1}(h)| \le [\log \frac{2^l}{|G|}] + d + \log(1 + \frac{|G|^{\frac{3}{4}}}{2^{\frac{1}{4}l}})$$

If G meets the following requirements $0 < d + \log(1 - \frac{|G|^{\frac{3}{4}}}{2^{\frac{1}{4}l}}) < d + \log(1 + \frac{|G|^{\frac{3}{4}}}{2^{\frac{1}{4}l}}) < 1$, then

$$\lceil \log |f^{-1}(h)| \rceil = \lceil \log \frac{2^l}{|G|} \rceil,$$

also

$$\forall h \in G, \lfloor \log |f^{-1}(h)| \rfloor = \lfloor \log \frac{2^l}{|G|} \rfloor.$$

□

It is necessary to point out that the restriction is easy to meet when l choose from some proper value in $O(n \log n)$.

Now we describe how to generate the abelian group G. Indeed what we generate, the abelian group, is a lattice $G = \mathbb{Z}_q^n$. It is generated as a random lattice.

The parameter q has to be chosen carefully, because $|G| = q^n$. That is to say, q should make $|G|$ meet the restriction in lemma 2. After we generate G, we randomly pick m elements of G to generate $\mathbf{A}$. And the one way function f is

$$f_{\mathbf{A}}(x) = \mathbf{A}x, x \in \{0,1\}^m.$$

The next theorem is in [HILL99] Theorem 5.1.4. The construction method used in [HILL99] could be used by applying ISSS to construct a PRG, but it requires the length of the seed to be $O(n^4)$ which is not optimal.

Theorem 1. *A pseudorandom generator can be constructed from a one-way function* f *if* $\lceil \log |f^{-1}| \rceil$ *could be computed in polynomial time. The reduction is weak-preserving.*

3.2 The Hardness of ISSS

As ISSS is a variation of ISIS, it has the same hardness property as ISIS does. In deed, each randomly generated input instance for a ISIS$_{q,m,\beta}$ oracle (with $\beta = \sqrt{m}$), could directly be used as a input for ISSS$_{q,m}$ oracle, where the output of the ISSS$_{q,m}$ oracle is exacly a proper result for the input ISIS$_{q,m,\beta}$ instance. ISIS$_{q,m,\beta}$ as proved in [MR07], is hard in average-case assuming GapSVP$_\gamma$ is hard, for a proper γ. The next theorem gives exactly the hardness result of ISSS.

Theorem 2. *For any $m = poly(n)$, $q \geq n^{2.5}\log n$, there exists $\gamma = O(n\sqrt{\log n})$ such that, solving $ISSS_{q,m}$ on the average case is at least as hard as solving $GapSVP_\gamma$ in the worst case.*

As a result, assuming GapSVP$_{O(n\log n)}$ is hard on the worst case, ISSS is hard on the average case. This means we could generate a hard instance of ISSS in polynomial time by random generation. So the function $f_{\mathbf{A}}$ we generated above is truly hard as a one-way function.

4 From One Way Function of ISSS to Pseudorandom Generator

In this section, we briefly describe how to construct a pseudorandom generator from our one-way function, using the method in [GKL93].

4.1 The Construction

The method in [GKL93] focus on how to construct a one way function on iterates. Our one way function also meets their requirements. Also we make a little improvement to make the seed shorter.

Now we describe the construction.

Construction 1. *Let $\mathcal{H}(m)$ be a universal class of hash functions. Each function in $\mathcal{H}(m)$ maps m-bit strings to m-bit strings.*

$$f'(h_0,\ldots,h_{t(m)-1},i,x) = (h_0,\ldots,h_{t(m)-1},i',h_i(f(x)))$$

where $x \in \{0,1\}^m, h_k \in \mathcal{H}(m), 0 \leq i \leq t(m)-1, i' = (i+1) \mod t(m), t(m) \geq m+1$.

Denote $\mathfrak{G}^{(i)}_{\mathcal{H}(m)}(f)(x) = h_i(\mathfrak{G}_{i-1}(f)(x))$, where $\mathfrak{G}^{(0)}(f)(x) = x$.

Definition 7. *Let $\mathcal{H}$ be a finite collection of hash functions mapping set I to set $O, m = |O|$. Then $\mathcal{H}$ is called universal if, given $x, y \in U, (x \neq y)$,*

$$|\{h \in \mathcal{H} : h(x) = h(y)\}| = \frac{|\mathcal{H}|}{m}$$

.

The following theorem is in [GKL93], section 2.6.

Theorem 3. *If f is a strong one-way function, $\forall y_1 = f(x_1), y_2 = f(x_2), \lfloor|f^{-1}(y_1)| = |f^{-1}(y_2)|\rfloor$, then in the construction above, f' is a one-way function on iterates.*

As the one-way function $f_{\mathbf{A}}(x)$ defined in the section 3 is a strong one-way function meets the requirements according to lemma 2, it could be used by the above method. The result is the following theorem.

Theorem 4. *If $f = f_{\mathbf{A}}$, f' constructed in construction 1 is a one-way function on iterates for $|x|$ steps.*

The problem is, the construction method gives in [GKL93] requires generating m hash functions from the universal class of hash function $\mathcal{H}$. Each hash function will require m^2 bits random bits to construct. It is easy to see that we need more than m^3 random bits to start our generator.

4.2 Optimization

Next we explains the method to decrease the seed length of the pseudorandom generator.

We will use toeplitz matrix to construct hash functions. This only require cm^2 random bits, where c is a constant.

In linear algebra, a Toeplitz matrix or diagonal-constant matrix, named after Otto Toeplitz, is a matrix in which each descending diagonal from left to right is constant. It is proved in [Krawczyk95] that toeplitz hash function is one kind of universal hash function. Each universal hash function is generated as the follows.

Construction 2. *Each universal hash function is of the form $h(x) = Tx$. T is a toeplitz matrix. Each of its main diagonal elements is randomly chosen from $\{0, 1\}$.*

A direct way to construct universal hash function $h(x) = Tx$ is to generate the bit matrix T, which need m^2 random bits, where m is the dimension of the matrix. Using toeplitz matrix it only need $2^m - 1$ random bits (exactly for the $2^m - 1$ diagonal elements). In this way, the overall random bits needed for m hash functions are $2m^2 - m$ in $O(m^2)$.

4.3 From one-way function on iterates to PRG

Former researcher had already provided method to construct PRG from one-way function on iterates.

There are two steps. First step is to construct a generator stretching 1 bit from the seed (1 bit generator). Second Step is to stretch many bits using the 1 bit generator. We summarize the methods as the following 2 theorem which could be concluded from [GKL93]. We just modified the number of random hash functions so that it is as less as possible. It could be easily verified that this modification do not affect its validity.

Theorem 5. *The 1 bit generator is as follows.*

$$\mathcal{F}(\mathcal{H}, x, r) = (f^{(m)}(x) \odot r, f^{(m-1)}(x) \odot r, \ldots, x \odot r, \mathcal{H}, r)$$

The input seed is $\mathcal{H}, x, r$, and the value of the function is the generated pseudorandom bit string. Here $\mathcal{H}$ represents the collection of hash function we randomly generated.

In our construction, we have $f^{(i)}(x) = \mathfrak{G}^{(i)}_{\mathcal{H}(m)}(f)(x)$. As a result, in order to generate m hash functions, $(2m - 1)m$ random bits are enough. Finally, the overall length of the seed is $2m^2$.

The last theorem is stated in [HILL99], Proposition 3.3.4.

Theorem 6. *If $g : \{0, 1\}^n \to \{0, 1\}^{n+1}$ is a pseudorandom generator that stretches by one bit. Define $g^{(1)}(x) = g(x)$, and inductively, for all $i \geq 1$,*

$$g^{(i+1)}(x) = \langle g(g^{(i)}(x)_{\{1,\ldots,n\}}), g^{(i)}(x)_{\{n+1,\ldots\ldots,n+i\}} \rangle.$$

Let k_n be a integer valued parameter which is computable in polynomial time of n. $g^{(k_n)}$ is a pseudorandom generator.

According to construction 1, theorem 4, 5 and 6, we use $f_{\mathbf{A}}$ as the one-function. Construct one-way function on iteration by construction 1. Use the method in theorem 5 to construct a 1 bit PRG. At last apply the method in theorem 5 to construct PRG which could generate arbitrary long (in polynomial of seed length) of pseudorandom bits.

5 Conclusions and Open Problems

According to our method, we could use lattice one-way function to generate a pseudorandom generator. Its hardness is based on the assumed hardness of $\text{GapSVP}_{O(n \log n)}$. The generator requires $O(m^2)$ bits as the seed, where m is in $O(n \log n)$.

However there are still some problems about our one-way function. In fact, intuitively, the ISSS problem could have a better hardness result. So it is a open problem whether ISSS could be proved hard based on harder problem (harder than $\text{GapSVP}_{O(n \log n)}$).

Another open problem is whether there are ways to make the required seed length shorter. It will be a better result if the needed seed length is in $O(n)$.

References

[Ajtai96] M. Ajtai. Generating hard instances of lattice problems. STOC, 1996, 99-108.

[AD96] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. STOC, 1997.

[Ajtai99] M. Ajtai. Generating Hard Instances of the Short Basis Problem. ICALP 99, LNCS 1644, pp. 1-9, 1999.

[Aharonov Regev04] Dorit Aharonov, Oded Regev. Lattice Problems in NP cap coNP. FOCS 2004: 362-371

[AP09] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. Theory of Computing Systems, 48(3): 535-553, April 2011. Preliminary version in STACS 2009.

[Arora Barak09] Sanjeev Arora, Boaz Barak. Computational Complexity, A Modern Approach. Cambridge University Express, ISBN-13: 978-0-511-53381-5, 2009.

[BM82] Blum, M., and Micali, S., "How to Generate Cryptographically Strong Sequences of Pseudorandom Bits", SIAM J. on Computing, Vol. 13, 1984, pp. 850-864.

[Babai86] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica, 1986.

[BH89] Boppana, R., Hirschfeld, R.. Pseudo-random generators and complexity classes. S. Micali, ed.. Advances in Computer Research, vol. 5. pp. 1-26, JAI Press, 1989.

[Banaszczyk93] W.Banaszczyk. New bounds in some transference theorems in the geometry of numbers. Mathematische Annalen, 1993.

[BLPRS13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, Damien Stehle. Classical Hardness of Learning with Errors. eprint, 2013.

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In STOC, pages 25-32. ACM, 15-17 May 1989.

[GKL93] Goldreich, O., Krawczyk, H. and Luby, M.. "On the existence of Pseudorandom Generators". SIAM J. on Computing, Vol. 22, No. 6, December, 1993, pp. 1163-1175.

[GM02] S. Goldwasser and D. Micciancio. Complexity of lattice problems. Springer, 2002.

[GPV08] C. Gentry, C. Peikert and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In STOC, pages 197-206. 2008.

[HILL99] J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. SIAM J. Comput., 28(4): 1364-1396, 1999.

[Krawczyk95] Hugo Krawczyk, "New Hash Functions for Message Authentication", EUROCRYPT, 1995.

[LLL1982] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. Mathematische Annalen, 261:513-534, 1982.

[Levin87] Levin, L.A., "One-way Function and Pseudorandom Generators", Combinatorica, Vol. 7, No. 4, 1987, pp. 357-363.

[Micciancio01] D. Micciancio. The shortest vector problem is **NP**-hard to approximate within some constant. SIAM journal on Computing, 2001, 30(6), 2008-2035.

[MR07] D. Micciancio and O. Regev. Worst-case to Average-case Reductions based on Gaussian Measures. FOCS, 2004.

[MV2010] Daniele Micciancio, Panagiotis Voulgaris. A Deterministic Single Exponential Time Algorithm for Most Lattice Problems based on Voronoi Cell Computations. STOC, 2010.

[MP2011] D. Micciancio, Chris Peikert. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. 2011.

[NV08] P. Nguyen and T. Vidick. Sieve algorithms for the shortest vector problem are practical. J. of Mathe-matical Cryptology, 2(2):181-207, jul 2008.

[NP08] Nicolas Gama, Phong Q. Nguyen. Finding Short Lattice Vectors within Mordell's inequality. 2008.

[Peikert09] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. STOC, 2009.

[Regev05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. J. ACM, 2009.

[Yao82a] A. C. C. Yao. Theory and applications of trapdoor functions. In FOCS, pages 80-91. IEEE, 3-5 Nov. 1982.