Theoretical Toys

Xiao Liang
https://xiao-liang.com

(Updated on May 23, 2021)

Chapter 1

Approximation

Some Notations. We use $f(x) \sim g(x)$ to denote the fact that $f(x) = g(x)(1 + o(1))$.

1.1 The Uncommon Friends of Big O

1.1.1 The Soft O

The soft-O notation $\widetilde{O}(n)$ is a variant of the big-O that "ignores" logarithmic factors. Formally, $f(n) \in \widetilde{O}\big(g(n)\big)$ if and only if there exists a constant k s.t. $O\big(g(n) \log^k g(n)\big)$.

It is worth noting that for any constant k and any ϵ, $\log^k(n) \in O(n^\epsilon)$. Therefore, the soft-O notation is often used to obviate the "nitpicking" within growth-rates that are stated as too tightly bounded for the matters at hand.

1.1.2 Sub-Exponential Time and Complexity Leveraging

The precise definition of "sub-exponential" is not generally agreed upon[1].

Definition 1.1.1: Sub-Exponential Time

Following are two most widely used definition for sub-exponential time:

1. Function $f(n)$ is sub-exponential if $f(n) \in O(2^{n^\epsilon})$ for every $\epsilon > 0$.
2. **(Cryptographers'.)** Function $f(n)$ is sub-exponential if $f(n) \in 2^{o(n)}$.

The first one is used in the olden literature of complexity theory, e.g., [BFNW93, Mil01]. The second formalism is more modern, e.g., [IP01, Reg04, Kup05]. Cryptographers seem to prefer the second one. Indeed, the second one captures the running time of the fastest known classical factoring algorithm (as well as that of the fastest known algorithm for graph isomorphism).

Complexity leveraging is a useful technique in cryptography [CGGM00]. It relies on sub-exponential assumptions (the second one in Definition 1.1.1). This technique is best demonstrated by the construction of 2-round SPS ZK protocol from [Pas04, Section 3.5].[2] (Give an overview of this example?)

[1]See this blog post by Scott Aaronson

[2]This is the same construction in [Pas03, Section 5]

1.1.3 The Iterated Logarithm

The iterated logarithm $\log^*(n)$ can be defined by the following recursive formula:

$$\log^*(n) := \begin{cases} 0 & n \leq 1 \\ 1 + \log^*\big(\log^* n\big) & n > 1 \end{cases}.$$

Intuitively, $y = \log^*(n)$ denotes the number of times the logarithm function must be *iteratively* applied to n before the result is ≤ 1. That is,

$$\underbrace{b^{b^{\cdot^{\cdot^{b}}}}}_{y-1 \text{ times}} \leq n \leq \underbrace{b^{b^{\cdot^{\cdot^{b}}}}}_{y \text{ times}}.$$

(Talk about [Wee10] as an example.)

1.2 Useful Asymptotics

1.2.1 Harmonic Numbers

Harmonic number is defined as $H_n = \sum_{i=0}^{n} \frac{1}{i}$. The following exact bound can be proved by the "integral trick".

$$\ln(n+1) \leq H_n \leq \ln(n) + 1.$$

This also implies that H_n is approximately $\ln(n)$, i.e. $H_n \sim \ln(n)$.

The proof of the following fact is left as a simple exercise [Hint: collecting adjacent items in a "binary fashion"]:

$$\lfloor \log n \rfloor + 1 \leq H_n \leq \frac{1}{2}\lceil \log n \rceil + 1$$

The main take-away is: $H_n = \Theta\big(\ln(n)\big)$.

1.2.2 Some Asymptotics from Taylor Series

Let us recall the following Maclaurin Series (Taylor expansion at the origin point $a = 0$) with some interesting implications (since we are talking about Maclaurin series, imagine that x is very close to 0 in the following):

- $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots$. (It converges for $x \in (-1, 1]$). This implies that $\ln(1+x) \sim x$ when $x \to 0$.

- $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$. (It converges for all $x \in \mathbb{R}$). This implies that $e^x \sim 1 + x$ when $x \to 0$. A quick way to remember this is: this is the exponential version of the above $\ln(1+x) \sim x$.

- $\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots$. (It converges for $x \in (-1, 1)$). This implies that $\frac{1}{1-x} \sim 1 + x$ when $x \to 0$.

These examples show how we can get helpful Computer-Science asymptotics from Maclaurin series. More Maclaurin expansion can be found at this Wikipedia page. In the following, we states more useful asymptotics obtained by this approach:

- $\frac{1}{1-\varepsilon} = 1 + \varepsilon \pm O(\varepsilon^2)$
- $(1+\varepsilon)^{\frac{1}{2}} = 1 + \frac{1}{2}\varepsilon \pm O(\varepsilon^2)$

Remark 1.2.1: On the Usage of Big-O

Note that the above use of Big-O notations is different from the standard usage that captures the behavior of an increasing function when x goes to infinity (called "Infinite Asymptotics"). Instead, it is used here to describe a decreasing function on a variable x approaching 0. Such an usage is called "Infinitesimal Asymptotics". See this Wikipedia page for an explanation. We remark that both usages can be unified under the same formal definition of the Big-O notation (via the limit superior).

1.2.3 Stirling Formula

We want to study the asymptotic behavior of $n!$. We start with the following simply approach.

Taking the logarithm of it and applying the "integral trick" give us the following sharp bounds:

$$n\ln(n) - n + 1 \quad \leq \quad \ln(n!) \quad \leq \quad n\ln(n) - n + 1 + \frac{1}{2}\ln(n), \tag{1.1}$$

where the upper bound requires the clever trick that we collect the extra triangle remainders above the $\ln(n)$ curve to a rectangle that is parallel to y-axis.

Equation (1.1) immediately implies the following sharp bounds:

$$\left(\frac{n}{e}\right)^n e \quad \leq \quad n! \quad \leq \quad \left(\frac{n}{e}\right)^n e\sqrt{n} \tag{1.2}$$

Equation (1.2) also implies:

$$n! = \widetilde{\Theta}\left(\left(\frac{n}{e}\right)^n\right).$$

This result is already very close to the ground truth. Actually, we can show

$$n! = \Theta\left(\left(\frac{n}{e}\right)^n \sqrt{n}\right),$$

by proving that the size of the slivers we dropped in the derivation of the upper bound in Equation (1.1) actually converges to some constant.

To do ...

Prove Stirling's formula:

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \tag{1.3}$$

More exactly, it is

$$n! = \sqrt{2\pi n}(\frac{n}{e})^n\Big(1+O(\frac{1}{n})\Big). \tag{1.4}$$

1.3 Bounds for Binomial Coefficients

Useful Equalities for Binomial Coefficients. We first presents a set of widely used equalities regarding binomial coefficients. For all integers n, k, and t such that the following terms are well-defined, we have:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \tag{1.5}$$

$$\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1} \tag{1.6}$$

$$\binom{n}{k}\binom{n-k}{t} = \binom{n}{t}\binom{n-t}{k} \tag{1.7}$$

The Deathbed Formula. Even if someone asks you about this formula on your deathbed, you should be able to spell it out without thinking.

$$(\frac{n}{k})^k \le \binom{n}{k} \le \frac{n^k}{k!} \le (\frac{n\cdot e}{k})^k \tag{1.8}$$

Subsets Non-Overlapping. Another useful bound that appears again and again in cryptographic applications is the following one:

Lemma 1.3.1: Subsets Non-Overlapping

Let $k < n$ and $t < (n-k)$. Then, we have

$$\frac{\binom{n-k}{t}}{\binom{n}{t}} \le \left(1-\frac{k}{n}\right)^t \text{ and } \frac{\binom{n-k}{t}}{\binom{n}{t}} \le \left(1-\frac{t}{n}\right)^k \tag{1.9}$$

Proof. The proof of Inequality (1.9) is rather simple:

$$\begin{aligned}
\frac{\binom{n-k}{t}}{\binom{n}{t}} &= \frac{(n-k)!}{t!(n-k-t)!}\frac{(n-t)!t!}{n!} = \frac{(n-k)!}{(n-k-t)!}\frac{(n-t)!}{n!} \\
&= (n-k)(n-k-1)\cdots(n-k-t+1)\cdot\frac{1}{n(n-1)\cdots(n-t+1)} \\
&= \frac{n-k}{n}\cdot\frac{n-k-1}{n-1}\cdots\frac{n-k-t+1}{n-t+1} = \left(1-\frac{k}{n}\right)\cdot\left(1-\frac{k}{n-1}\right)\cdots\left(1-\frac{k}{n-t+1}\right) \\
&\le \left(1-\frac{k}{n}\right)^t
\end{aligned}$$

Note that Equation (1.7) essentially says that the role of k and t are interchangeable in the fraction considered above. Thus, the above result together with Equation (1.7) gives us the second part of Inequality (1.9). ■

We show the following simple corollary as an example of the application of Lemma 1.3.1.

Corollary 1.3.2: Subset-Guessing Game

Let $n(\lambda)$ be a polynomial. Let $k(\lambda) = \delta n(\lambda)$ where $0 < \delta < 1$ is a constant. Let $t(\lambda) = \omega(\log \lambda)$ and $t(\lambda) < n(\lambda) - k(\lambda)$. For any computationally-binding commitment scheme Com, no PPT adversary Adv can win the following "subset-guessing" game with non-negligible probability:

1. A challenger samples a random size-t subset $\mathsf{r} = \{b_1, \ldots, b_t\} \subseteq [n]$, and commits to this subset to Adv using Com;
2. Adv then outputs a size-k subset $\{p_1, \ldots, p_k\} \subseteq [n]$;
3. The Adv wins if $\{b_1, \ldots, b_t\} \subset [n] \setminus \{p_1, \ldots, p_k\}$.

Proof. Assume for contradiction that there is a computationally-hiding Com and a PPT Adv that wins in the above game with non-negligible probability. We then show a PPT machine Adv_h that breaks the computationally-hiding property of Com:

1. Adv_h samples independently two random size-t subsets of $[n]$, denoted as $B = \{b_1, \ldots, b_t\}$ and $B' = \{b'_1, \ldots, b'_t\}$. Adv_h sends B_0 and B_1 to the external challenger for the hiding game of Com;

2. Adv_h then internally invokes Adv and relay messages between Adv and the external challenger;

3. After the interaction with the external challenger, Adv will output a set $\{p_1, \ldots, p_k\}$. Adv_h output 1 if and only if $B \subseteq [n] \setminus \{p_1, \ldots, p_k\}$.

In the following, we argue that the following value is non-negligible, which means that Adv_h breaks the hiding of Com:

$$\left|\Pr[\mathsf{Adv}_h = 1 \mid \mathsf{Com}(B)] - \Pr\big[\mathsf{Adv}_h = 1 \mid \mathsf{Com}(B')\big]\right|.$$

First, note that Adv's view in the above game is identical to that in the subset guessing game. It then follows from our assumption that $\Pr[\mathsf{Adv}_h = 1 \mid \mathsf{Com}(B)]$ is non-negligible. Therefore, it suffices to show that $\Pr[\mathsf{Adv}_h = 1 \mid \mathsf{Com}(B')]$ is negligible. Recall that $\mathsf{Alice}dv_h$ outputs 1 if and only if $B \subseteq [n] \setminus \{p_1, \ldots, p_k\}$. However, conditioned on $\mathsf{Com}(B')$ (i.e. the external challenger commits to B'), Adv has no information about B. Thus, $\{p_1, \ldots, p_k\}$ and B are independently distributed. We then have:

$$\Pr\big[\mathsf{Adv}_h = 1 \mid \mathsf{Com}(B')\big] = \frac{\binom{n-k}{t}}{\binom{n}{t}} \le \left(1 - \frac{k}{n}\right)^t = (1-\delta)^t \tag{1.10}$$

By our choice of parameter, $0 < \delta < 1$ is a constant and $t = \omega(\lambda)$. Therefore, $\Pr[\mathsf{Adv}_h = 1 \mid \mathsf{Com}(B')] = \mathsf{negl}(\lambda)$. This finishes the proof of Corollary 1.3.2. ■

Chapter 2

Algebra from a Modern Point of View

2.1 Pre-Group Concepts

Definition 2.1.1: Magma

A magma (also called "groupoid") is a set M equipped with a binary operation "+" satisfying the following property:

1. **Closure.** M is closed under "+".

Definition 2.1.2: Semigroup

A semigroup is a set S equipped with a binary operation "+" satisfying the following properties:

1. **Closure.** S is closed under "+".
2. **Associativity.** For all $a, b, c \in S$, $(a + b) + c = a + (b + c)$.

Definition 2.1.3: Monoid

A monoid is a set M equipped with a binary operation "+" satisfying the following properties:

1. **Closure.** M is closed under "+".
2. **Associativity.** For all $a, b, c \in M$, $(a + b) + c = a + (b + c)$.
3. **Identity Element.** There is an element e in M such that for all $a \in M$, $a + e = e + a = a$.

The relations among these concepts can be summarized as follows:

- A magma is the most basic algebraic structure (over a set).
- A semigroup is a magma with associativity.
- A monoid is a semigroup with an identity element.

2.2 Groups

Definition 2.2.1: Group

A group is a set G equipped with a binary operation "+" satisfying the following properties:

1. **Closure.** G is closed under "+".
2. **Associativity.** For all $a, b, c \in G$, $(a + b) + c = a + (b + c)$.
3. **Identity Element.** There is an element e in G such that for all $a \in G$,
$$a + e = e + a = a$$
4. **Inverse Element.** For any $a \in G$, there is an element $-a$ in G such that
$$a + (-a) = (-a) + a = e$$

A group is called "Abelian" if it additionally satisfies the following property

5. **Commutativity.** For all $a, b \in G$, $a + b = b + a$.

(Talk about the relation between Branching Program and Symmetric groups)

(Some book uses "factor group" to refer to "quotient group". They are the same.)

Here is a simple (but very useful) fact of finite group. It gives Euler's theorem when instantiated on group $\mathbb{Z}_n^*$. (The proof is omitted as it is obvious.)

Theorem 2.2.2:

Let G be a finite group of order $m = |G|$. Then $\forall g \in G$, $g^m = 1$.
Specifically, if we set $G = \mathbb{Z}_n^*$ $(n \in \mathbb{N})$, this is the Euler's theorem:

$$\forall a \in \mathbb{Z}_n^*, \quad a^{\phi(n)} = 1 \bmod n.$$

Theorem 2.2.2 gives the following two very important corollaries. The first one is extremely useful for cryptography as it tells a sufficient condition to construct permutation on finite groups. The second one is helpful to compute large exponentiation on finite groups.

Corollary 2.2.1. Let G be a finite group of order $m > 1$. Let $e > 0$ be an integer, and define the function $f_e : G \to G$ by $f_e(g) = g^e$. We have:

$$\mathsf{gcd}(e, m) = 1 \quad \Rightarrow \quad f_e \text{ is bijective}$$

Moreover, if $d = e^{-1} \bmod m$ then f_d is the inverse of f_e.

$\Diamond$

Corollary 2.2.2. Let G be a finite group of order $m > 1$. Then for any $g \in G$ and any integer x,we have $g^x = g^{x \bmod m}$.

$\Diamond$

Other interesting corollaries of Theorem 2.2.2 include:

- Let G be a finite group, and $g \in G$ an element of order i. Then:

$$g^x = g^y \quad \Leftrightarrow \quad x = y \bmod i$$

- Let G be a finite group of order m, and say $g \in G$ has order i. Then $i|m$.

2.2.1 Cyclic Groups

Cyclic groups are a type of groups that is of special interest for cryptographers. Several number-theoretic problems are conjectured to be intractable on cyclic groups, while there do exist some non-cyclic groups where these problems are easy.

The first fact we what to stress is that every finite group of prime order is cyclic. This can be regarded as another corollary of Theorem 2.2.2.

Theorem 2.2.3:

If G is a group of prime order p, then G is cyclic. Furthermore, all elements of G except the identity are generators of G.

The following theorem is very important. It shows that $\mathbb{Z}_p^*$ is a cyclic group if p is a prime. Note that is does not follow as a corollary of Theorem 2.2.3. Actually, its proof is very involved but can be found in standard abstract algebra textbooks.

Theorem 2.2.4:

If p is prime then $\mathbb{Z}_p^*$ is a cyclic group of order $p-1$.

Why does cryptography prefer cyclic groups?

- A cyclic group can be described by a single generator. Also, every element is a generator.

In addition, cyclic groups of *prime order* enjoy additional advantages[1]:

- This is a consequence of the Pohlig–Hellman algorithm, described in Chapter 9, which shows that the discrete-logarithm problem in a group of order q becomes easier if q has (small) prime factors. This does not necessarily mean that the discrete-logarithm problem is easy in groups of nonprime order; it merely means that the problem becomes easier.

- Related to the above, DDH problem is easy if the group order q has small prime factors. For example, in group $\mathbb{Z}_p^*$ with p a prime, discrete log is believed to be hard, but DDH is usually easy. Thus, people have to use subgroups of $\mathbb{Z}_p^*$ of prime order for DDH-based constructions (see Theorem 2.2.5).

- Finding a generator in cyclic groups of prime order is trivial. In contrast, efficiently finding a generator of an arbitrary cyclic group requires the factorization of the group order to be known (see Appendix B.3 of [KL14]).

[1]These are the reasons listed in [KL14]

- When the group order is prime, any nonzero exponent will be invertible, making this computation of multiplicative inverses possible.
- Consider the DDH tuple (g^a, g^b, g^{ab}). For it to be indistinguishable form a random tuple, a necessary is that g^{ab} by itself should be indistinguishable from a uniform group element. One can show that g^{ab} is "close" to uniform (in a sense we do not define here) when the group order p is prime, something that is not true otherwise.

We present a useful theorem w.r.t. the form of subgroups of $\mathbb{Z}_p^*$.

Theorem 2.2.5:

Let $p = rq + 1$ with p, q prime. Then $G := \{h^r \bmod p \mid h \in \mathbb{Z}_p^*\}$ is a subgroup of $\mathbb{Z}_p^*$ of order q.

2.2.2 $\mathbb{Z}_N$, $\mathbb{Z}_N^*$ and RSA

Lemma 2.2.3. Let $a \geq 1$, $n > 1$ be integers. Then a is invertible in $\mathbb{Z}_n$ if and only if $\mathsf{gcd}(a, n) = 1$.

$\diamondsuit$

The RSA Assumption. We first define a set of all integers when are the product of two length-λ primes:

$$Z_\lambda^{(2)} = \{N \mid N = p \cdot q \text{ where } p \text{ and } q \text{ are } \lambda\text{-bit primes.}\}$$

The RAS assumption conjunctures that the following problem is hard: for $N \overset{\$}{\leftarrow} Z_\lambda^{(2)}$, e such that $\mathsf{gcd}(e, \phi(N)) = 1$[2] and $y \overset{\$}{\leftarrow} \mathbb{Z}_N^*$, the computational task the adversary Adv is to find x such that $x^e = y \bmod N$. The (n, t, ε) hardness of RSA assumption is: no t-time algorithm Adv satisfies:

$$\Pr[\mathsf{Adv}(N, e, y) = x \text{ where } x^e = y \bmod N] > \varepsilon$$

Further discussion regarding the choice of e and other parameters can be found in [KL14].

2.2.3 Quadratic Residuosity

Legendre Symbol and Jacob Symbol.

Definition 2.2.6: Legendre Symbol

Let p be an odd prime. The Legendre symbol of an integer a is defined as

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & a \text{ is a QR and } a \neq 0 \bmod p \\ -1 & a \text{ is a QNR} \\ 0 & a = 0 \bmod p \end{cases}.$$

[2]This requirement is to guarantee that e induces a permutation on $\mathbb{Z}_N^*$ (see Corollary 2.2.1) such that the RAS problem is well defined. Namely, every y has a preimage under $f_e(x) = x^e \bmod N$.

Lemma 2.2.4. Let p be an odd prime. Then $\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}}$. ◊

Definition 2.2.7: Jacobi Symbol

Let N be a positive odd integer. The Jacobi symbol of an integer a is defined as

$$\mathcal{J}_N(a) \coloneqq \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{\alpha_i} = \left(\frac{a}{p_1}\right)^{\alpha_1} \cdot \left(\frac{a}{p_2}\right)^{\alpha_2} \cdots \left(\frac{a}{p_k}\right)^{\alpha_k},$$

where $N = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$.

(Through Section 2.2.3, we define $N = pq$, where p and q are primes of equal length.)

A tentative outline:

- By Chinese remainder theorem, $\mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*$. Denote the isomorphism as $y \leftrightarrow (y_p, y_q)$.
- For $y \in \mathbb{Z}_N^*$ and $y \leftrightarrow (y_p, y_q)$, it can be proved that y is a QR in $\mathbb{Z}_N^*$ if and only if y_p is a QR in $\mathbb{Z}_p^*$ and y_q is a QR in $\mathbb{Z}_q^*$.
- The above implies: each QR $y \in \mathbb{Z}_N^*$ has exactly four square roots.
- Let QR_N set of quadratic residues modulo N. Let QNR_N set of quadratic non-residues modulo N. We have

$$\frac{|\mathsf{QR}_N|}{|\mathbb{Z}_N^*|} = \frac{|\mathsf{QR}_p| \cdot |\mathsf{QR}_q|}{|\mathbb{Z}_N^*|} = \frac{\frac{p-1}{2} \cdot \frac{q-1}{2}}{(p-1)(q-1)} = \frac{1}{4}.$$

  Note that since $\mathbb{Z}_p^*$ is cyclic, we can easily show that $|\mathsf{QR}_p| = \frac{p-1}{2}$, i.e. half of the elements in $\mathbb{Z}_p^*$ are QRs.
- Also, for $x, y \in \mathbb{Z}_N^*$, we have

$$\mathcal{J}_N(x \cdot y) = \mathcal{J}_N(x) \cdot \mathcal{J}_N(y) = \mathcal{J}_p(x) \cdot \mathcal{J}_q(x) \cdot \mathcal{J}_p(y) \cdot \mathcal{J}_q(y).$$

- Let $\mathcal{J}_N^+$ (resp. $\mathcal{J}_N^-$) denote the set of elements in $\mathbb{Z}_N^*$ whose Jacobi symbol is $+1$ (resp. -1). Let QNR_N^+ denote the set of elements in QNR_N whose Jacobi symbol is $+1$. Then we can show the follows:
  - $\mathbb{Z}_N^* = \mathcal{J}_N^- \cup \mathcal{J}_N^+$ and $|\mathcal{J}_N^-| = |\mathcal{J}_N^+|$;
  - $\mathcal{J}_N^+ = \mathsf{QR}_N \cup \mathsf{QNR}_N^+$ and $|\mathsf{QR}_N| = |\mathsf{QNR}_N^+|$.
- Recall that when the factorization of N is unknown, there is no known polynomial-time algorithm for deciding whether a given x is QR or not. But, somewhat surprisingly, a polynomial-time algorithm is known for computing $\mathcal{J}_N(x)$ without the factorization of N.
- Quadratic residuosity assumption says that it is hard to tell between a random sample from QR and a random sample from QNR^+.

  Definition 2.2.5 (QR assumption)**.** Quadratic residuosity assumption assumes that there exists a generation algorithm Gen such that for all PPT algorithm Adv,

$$\left|\Pr[\mathsf{Adv}(N, \mathsf{qr}) = 1] - \Pr[\mathsf{Adv}(N, \mathsf{qnr}) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where the probabilities are taken over the following sampling $(N, p, q) \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathsf{qr} \xleftarrow{\$} \mathsf{QR}_N$ and $\mathsf{qnr} \xleftarrow{\$} \mathsf{QNR}_N^+$. $\Diamond$

2.3 Rings

Definition 2.3.1 (Ring)**.** A ring is a set R equipped with two binary operations "+" (usually called *addition*) and $\cdot$ (usually called *multiplication*) satisfying the following properties:

1. R is an Abelian group under "+".
2. R is a monoid under "$\cdot$".[3]
3. The multiplication is distributive with respect to the addition, meaning that:
   - (Left Distributivity) For all $a, b, c \in R$, $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.
   - (Right Distributivity) For all $a, b, c \in R$, $(b + c) \cdot a = (b \cdot a) + (c \cdot a)$.

$\Diamond$

Definition 2.3.2 (Ideal)**.** A subring A of a ring R is called a (two-sided) ideal of R if for every $r \in R$ and every $a \in A$, both ra and ar are in A. $\Diamond$

Theorem 2.3.3. If A is an ideal of a ring R, then the quotient group R/A is a ring under the following operation:

- **Addition.** $(s + A) + (t + A) = (s + t) + A$
- **Multiplication.** $(s + A) \cdot (t + A) = (s \cdot t) + A$

$\Diamond$

There is special type of ideals defined on commutative rings that we are interested in, especially when we talk about polynomial rings later. It is called principal ideal.

Definition 2.3.4 (Principal Ideal)**.** Let R be a commutative ring (with unity) and let $a \in R$. The set $\langle a \rangle = \{ra \mid r \in R\}$ is an ideal of R. We call it the principal ideal generated by a. $\Diamond$

[3] We remark that some mathematicians prefer to define the ring without multiplicative identity (the unity). So in their definition, R is a semigroup under "$\cdot$", instead of a monoid. But some other mathematicians prefer the current definition. We choose to use the current one because we almost always need the existence of unity. In this book, we put "(with unity)" wherever we want to address it.

2.4 Fields

Definition 2.4.1: Field

A field is a set F equipped with two binary operations "+" (usually called *addition*) and "·" (usually called *multiplication*) satisfying the following properties:

1. F is an Abelian group under the addition.
2. $F \setminus \{0\}$ form an Abelian group under the multiplication
3. The multiplication is distributive over the addition.

2.5 Modules and Vector Spaces

Definition 2.5.1: Modules

Let R be a ring (not necessarily with unity). A left (resp. right) R-module over R is a set M together with:

1. a binary operation "+" under which M is an Abelian group.
2. a map $R \times M \to M$ (resp. $M \times R \to M$) denoted by "·", such that for all $r, s \in R$ and $m, n \in M$ the following holds:
   (a) $(r + s) \cdot m = r \cdot m + s \cdot m$ (resp. $m \cdot (r + s) = m \cdot r + m \cdot s$)
   (b) $(rs) \cdot m = r \cdot (s \cdot m)$ (resp. $m \cdot (rs) = (m \cdot r) \cdot s$)
   (c) $r \cdot (m + n) = r \cdot m + r \cdot n$ (resp. $(m + n) \cdot r = m \cdot r + n \cdot r$)

   If R has an unity 1, we impose an additional axiom to the map:

   (d) $1 \cdot m = m$ (resp. $m \cdot 1 = m$)

A Remark on the terminology: A *bimodule* is a module that is a left module and a right module such that the two multiplications are compatible. If R is commutative, then left R-modules are the same as right R-modules and are simply called R-modules[4]. Note that Item (d) is optional; modules satisfying it are called unital modules.

One elegant application of modules in cryptography appears in the famous Groth-Sahai [GS08] proof systems. It is not because they use fancy theorems specific to modules; rather, the concept of modules provides a high-level abstract for groups equipped with bilinear maps, thus gives a clear and unified way to interpret their results.

Definition 2.5.2: Vector Spaces

Let $\mathbb{F}$ be a field. The $\mathbb{F}$-module is called a vector space over the field $\mathbb{F}$.

[4]To some authors, "R-module" by default means "left R-module", e.g. [DF04].

2.6 Integral Domains

We want to capture all the properties that integers enjoy. If we compare the definition of the ring to the set of integers, two important properties are missing: (1) commutativity and (2) cancellation property. Thus, people propose the concept of integral domain, which plays a prominent role in number theory and algebraic geometry.

Definition 2.6.1 (Unit)**.** we say that an element u of a ring R is a unit (also called "invertible element") if there is another element $v \in R$ such that $uv = vu = 1$. $\Diamond$

Definition 2.6.2 (Zero Divisors)**.** In a commutative ring R, $a \neq 0$ is a zero divisor if there is a nonzero element $b \in R$ such that $ab = 0$. $\Diamond$

Definition 2.6.3 (Integral Domain)**.** An integral domain is a commutative ring (with unity) that does not have zero divisors. $\Diamond$

Certain kinds of integral domain are of our interest. Next, we will list some related concepts and then study them in order.

Definition 2.6.4 (Association)**.** Elements a and b of an integral domain D are called associates if $a = ub$, where u is a unit of D. $\Diamond$

Definition 2.6.5 (Reducibility)**.** Let D be an integral domain. A non-zero, non-unit element a is called an irreducible if the following holds:

- whenever a is expressed as a product $a = bc$ with $b, c \in D$, then b or c is a unit.

A non-zero, non-unit element of D that is not irreducible is called reducible. $\Diamond$

Definition 2.6.6 (Primes)**.** In an integral domain, a non-zero, non-unit element a is called a prime if the following holds:

- $a|bc$ implies $a|b$ or $a|c$.

$\Diamond$

2.6.1 Principal Ideal Domain (PID)

Definition 2.6.7 (Principal Ideal Domain)**.** An integral domain D is called a principal ideal domain if every ideal of D has the form $\langle a \rangle$ for some $a \in D$. $\Diamond$

Exercise 2.6.8. Here are some simple exercises to help you get a familiar with these concepts.

(a) In an integral domain, every prime is an irreducible.

(b) In a PID, an element is an irreducible if and only if it is a prime.

2.6.2 Unique Factorization Domain (UFD)

We now have the necessary terminology to formalize the idea of unique factorization.

Definition 2.6.9 (Unique Factorization Domain)**.** An integral domain D is a unique factorization domain if the following holds:

1. every non-zero, non-unit element of D can be written as a product of irreducibles of D,
2. the factorization into irreducibles is unique up to associates and the order in which the factors appear.

$\Diamond$

2.6.3 Euclidean Domain (ED) and GCD Domain

Definition 2.6.10 (Euclidean Domain (ED))**.** An integral domain D is called a Euclidean domain if there is a function d (called the measure) from the nonzero elements of D to the nonnegative integers such that:

1. $d(a) \leq d(ab)$ for all nonzero $a, b \in D$
2. if $a, b \in D$ and $b \neq 0$, then there exist elements q and r in D such that $a = bq + r$, where $r = 0$ or $d(r) < d(b)$.

$\Diamond$

From the above definition, it is easy to see that in an ED, the Euclidean algorithm is well defined. Actually, we call it "Euclidean Domain" because it is the integral domain where we can run Euclidean algorithm to compute the unique GCD between any pair of elements.

But we remark that GCD can be defined without referring to Euclidean algorithm. Actually, there is a strictly super-set of ED, called GCD domain, where GCD is defined but may not be unique, and Euclidean algorithm is not admitted.

Definition 2.6.11 (Greatest Command Divisor (GCD))**.** Let R is a commutative ring. We say that $d \in R$ is a greatest common divisor (GCD) of $a, b \in R$ if the following two conditions are satisfied:

1. $d|a$ and $d|b$.
2. For any $c \in R$ with $c|a$ and $c|b$, we have $c|d$.

$\Diamond$

Definition 2.6.12 (GCD Domain)**.** An integral domain D is a GCD domain if for each pair of $a, b \in D \setminus \{0\}$, there exists a greatest common divisor. $\Diamond$

(Here is my intuition which needs to be verified: GCD in an ED must be unique. But GCD in a GCD domain is not necessarily unique (counter examples?).)

Theorem 2.6.13 (Relations among different types of Rings)**.** The relations among different types of rings can be summarized as follows:

$$\begin{aligned}&\text{ED} \subset \text{PID} \subset \text{UFD} \subset \text{GCD Domains} \subset \\ &\qquad \text{Integrally Closed Domains} \subset \text{ Integral Domains} \subset \text{Commutative Rings}\end{aligned}$$

Note that all the subset relations are proper. ◊

2.7 Polynomials

2.7.1 The Ring-Theory Definition

The abstract-algebraic interpretation of polynomials is to consider it as a special ring, i.e. the ring of polynomials. This is perhaps the most mathematically-correct approach to characterize polynomials.

An intuitive way to understand this interpretation is as follows: we start by adding an extra element x (called "indeterminate" or "variable") to a commutative[5] ring R. As we will see, it actually gives us a new ring, which we denote as $R[x]$. Let us consider the form of elements in $R[x]$. Because of the closure property of a ring, for any $a \in R$ and any $i \in \mathbb{N}$, ax^i should also be in $R[x]$, and so is their sum. Therefore, any expression of the form $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x^1 + a_0$ should be an element in $R[x]$. This reminds us of the concept of polynomials. Moreover, it is easy to prove that all elements of such a form do form a ring (i.e. all elements in $R[x]$ have such a form). Thus we name $R[x]$ as the "polynomial ring".

Definition 2.7.1 (Polynomial Ring)**.** Let R be a commutative ring. The set of formal symbols

$$R[x] = \{a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x^1 + a_0 \mid a_i \in R, n \text{ is a nonnegative integer}\}$$

forms a ring under he natural polynomial addition and multiplication operation, with the natural identity elements for addition and multiplication. ◊

Exercise 2.7.1

Here are some interesting exercises to reveal the relation between a polynomial ring and its underlying ring.

1. If D is an integral domain, then $D[x]$ is an Integral Domain.
2. If F is a field, then $F[x]$ is a Principal Ideal Domain.
3. If F is a field, then $F[x]$ is a Euclidean Domain (with the degree of polynomials as the Euclidean measure).

[5] If the ring is not commutative, we will need to distinguish between ax^2 and xax.

The reducibility concept of polynomials is just an instantiation of the reducibility of a standard integral domain on an ID of polynomials (see Def. 2.6.5).

Definition 2.7.2: Reducibility of Polynomials over an ID

Let D be an integral domain. A non-zero, non-unit element $f(x) \in D[x]$ is irreducible over D if the following holds:

- whenever $f(x)$ is expressed as a product $f(x) = g(x) \cdot h(x)$ with $g(x), h(x) \in D[x]$, then $g(x)$ or $h(x)$ is a unit in $D[x]$.

A non-zero, non-unit element of $D[x]$ that is not irreducible over D is called reducible over D.

2.7.2 Schwartz-Zipple lemma

A crypto application of Schwartz-Zipple can be found in [KOS18].

But this lemma is widely used in PCP theorem, sum-check protocols and property testing.

An excellent survey of this lemma can be found in this article by Lipton.

Theorem 2.7.3: Schwartz-Zipple Lemma

Suppose that $P(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ is a non-zero polynomial of total degree d over a field $\mathbb{F}$, and S is a non-empty subset of the $\mathbb{F}$. Then,

$$\Pr\left[P(x_1, \ldots, x_n) = 0\right] \leq \frac{d}{|S|}.$$

2.7.3 The Fundamental Theorem of Algebra

To do..

Add The Fundamental Theorem of Algebra here

2.7.4 On $\mathbb{F}_{p^n}$: An Application for [Sah99] NIZK

In [Sah99], Sahai used the number of roots of polynomials on finite fields to design a clever mechanism that enjoys the following property: for some parameter ℓ and t, it allows one to sample t (a fixed polynomial) sets of size ℓ, such that no $(t-1)$ sets out of these t sets cover the remaining one. This mechanism is essential to extend the famous [Sah99] non-malleable NIZK to support (bounded) multiple proofs.

(Add this application here. Abstract from [Sah99].)

2.7.5 Shamir's Secret Sharing

We start with the famous Lagrange's interpolation, which is an elegant method to find a polynomial that satisfies a bunch of points.

Algorithm 2.7.2 (Lagrange's Interpolation)**.** Given a set of $k+1$ data points:

$$(x_0, y_0), \dots, (x_i, y_i), \dots, (x_n, y_n)$$

where no two x_i's are the same, the interpolation polynomial in the Lagrange form is defined as:

$$L(x) = \sum_{i=0}^{n} y_i \cdot \ell_i(x) \tag{2.1}$$

where each ℓ_i is:

$$\ell_i(x) := \prod_{\substack{0 \le m \le k \\ m \ne i}} \frac{x - x_m}{x_i - x_m} = \frac{(x - x_0)}{(x_i - x_0)} \cdots \frac{(x - x_{i-1})}{(x_i - x_{i-1})} \frac{(x - x_{i+1})}{(x_i - x_{i+1})} \cdots \frac{(x - x_n)}{(x_i - x_n)} \tag{2.2}$$

We have $L(x_i) = y_i$ for all $i \in \{0, \dots, n\}$. And $L(x)$ is a polynomial of degree at most n.

Lagrange's interpolation can be generalized to any finite field, with the corresponding field operation.

Remark 2.7.4:

We remark that Lagrange's interpolation allow us to recover the whole polynomial express of $L(x)$. Actually, we can also recover $L(x^*)$ at a certain point x^*. To do that, just evaluate $\ell_i(x^*)$'s according to Eq. 2.2, and plug them into 2.1. This is a simple observation, but it turns to be very useful for building the Fuzzy IBE scheme in [SW05].

With the understanding of Lagrange's interpolation, we are know ready to present Shamir's Secrete Sharing scheme.

Algorithm 2.7.3 (Shamir's Secrete Sharing)**.** A t-out-of-n secrete sharing scheme can be constructed in the following way.

Given a finite filed $\mathbb{F}$, to share a secrete s:[6]

1. Choose $a_1, \dots a_{t-1} \overset{\$}{\leftarrow} \mathbb{F}$.
2. Define a polynomial $f(x) = s + a_1 x + \dots a_t x^{t-1}$.
3. Choose n distinct points $x_1, \dots, x_n \in \mathbb{F}$.
4. For $i \in [n]$, output $(x_i, f(x_i))$ as the secret share for party P_i.

[6] W.l.o.g., we assume $s \in \mathbb{F}$

When t or more parties try to recover the secrete, they can recover the polynomial $f(x)$ using Lagrange's interpolation, and then learn the secrete s from the constant term of $f(x)$.

2.7.6 Verifiable Secret Sharing

2.7.7 Fast Fourier Transform

(talk about it's application to efficient integer multiplication)

(See this YouTube playlist for an amazing series of talks on Fourier Transform (and Fourier analysis in general).)

Chapter 3

Linear Algebra for Quantum Information Theory

3.1 The Basics

To do...

The familiarity with the following topics represents minimal background requirements for quantum information theory. A great book for them is [Axl15].

- Define Hermitian matrix, positive semi-definite matrices. And talk about the eigenvalue decomposition of them.
- Spectral decomposition (aka eigen-decomposition) is the factorization of a matrix into a canonical form, whereby the matrix is represented in terms of its eigenvalues and eigenvectors. Only diagonalizable matrices can be factorized in this way. This decomposition captures the essence of density matrix (in quantum computing): every density matrix ρ (i.e. positive semi-definite matrix with trace 1) have a spectral decomposition, where eigenvalues are non-negative and the sum to 1, and the eigenvectors constitute orthonormal basis.
- Hilbert Space.
- Schmidt Decomposition. Define Schmidt decomposition and talk about its application in Uhlmann's Theorem. See this.

Here are some simple-yet-important facts about Kronecker product and other matrix operations, which appear repeatedly in Quantum computations.

- $\operatorname{tr}(A+B) = \operatorname{tr}(A) + \operatorname{tr}(B)$ and $\operatorname{tr}(A) = \operatorname{tr}(A^T)$.
- **Cyclic property of trace:** $\operatorname{tr}(ABC) = \operatorname{tr}(CAB) = \operatorname{tr}(BCA)$.
- $\operatorname{tr}(A \otimes B) = \operatorname{tr}(A)\operatorname{tr}(B)$. (Note that $\operatorname{tr}(AB) \neq \operatorname{tr}(A)\operatorname{tr}(B)$)
- $(AB)^* = A^*B^*$ and $(AB)^\dagger = B^\dagger A^\dagger$.
- $(A \otimes B)^* = A^* \otimes B^*$, $(A \otimes B)^T = A^T \otimes B^T$, and $(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$, where "$\dagger$" denotes Hermitian transpose (aka conjugate transpose).
- $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$, where AC and BD are meaningful matrix multiplications.

- $|\alpha_0\rangle\langle\alpha_1| \otimes |\beta_0\rangle\langle\beta_1| = (|\alpha_0\rangle \otimes |\beta_0\rangle)(\langle\alpha_1| \otimes \langle\beta_1|) = |\alpha_0\beta_0\rangle\langle\alpha_1\beta_1|$
- The Kronecker product operator "$\otimes$" is both *bilinear* and *associative*.
- Another way to state Hadamard gate: for $b \in \{0,1\}$, $H|b\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^b|1\rangle)$.
- A workhorse formula appeared in several quantum algorithms:

  $$H^{\otimes n}|x_1,\ldots,x_n\rangle = \sum_{y\in\{0,1\}^n} (-1)^{\langle x,y\rangle}|y_1,\ldots,y_n\rangle,$$

  where $x = (x_1,\ldots,x_n)$ and $y = (y_1,\ldots,y_n)$ are binary vectors.
- **Quantum Fourier Transform.** Let $x \in [N]$ where $N = 2^n$. Then N-dimensional Quantum Fourier Transform F_N is defined by the following unitary mapping:

  $$F_N|x\rangle \to \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1}\omega_N^{x\cdot y}|y\rangle,$$

  where ω_N is the N-th root of unity, i.e. $\omega_N = e^{\frac{2\pi i}{N}} = e^{-\frac{2\pi i}{N}}$.

3.1.1 Tensor Products of Hilbert Spaces

At the center of quantum computing lies tensor product of vector spaces. The pure-mathematical way to approach this concept involves formal discussion on tensor products of modules. Indeed, vector spaces are just a special type of modules. The reader can find comprehensive resources for this approach in [DF04, Chapter 10.4 and 11.5] and this lecture note by Prof. Conrad.

Fortunately, for quantum computing, we only need to focus on a very specific type of the above concept, i.e. tensor product of Hilbert spaces. Before throwing out the definition, let me first motivate it. We know that each isolated quantum system can be represent by a Hilbert space. Now what should we do if we want to describe two (or more) quantum systems? To do that, ideally, we hope to combine spaces V and W in a way that reserves all the good mathematical properties. For example, the resulted space would better be a vector space, which also admits inner product. This turns to be achievable exploiting the Kronecker product operation "$\otimes$".

Definition 3.1.1: Tensor Product of Hilbert Spaces

Let V and W be two Hilbert spaces. The tensor product of them is the vector space $V \otimes W$ whose elements are linear combinations of $|v\rangle \otimes |w\rangle$ where $|v\rangle \in V$, $|w\rangle \in W$ and $\otimes$ is the Kronecker product.

If is easy to check that the above definition is well-defined, i.e. $V \otimes W$ as defined above is indeed a vector space. We now state two important facts about $V \otimes W$:

- It can be proved that the linear operators on $V \otimes W$ are captured by matrix Kronecker product $\mathbf{A} \otimes \mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are linear operators on V and W respectively. Namely, for any $|v\rangle \in V$ and $|w\rangle \in W$,

$$(\mathbf{A} \otimes \mathbf{B})(|v\rangle \otimes |w\rangle) = \mathbf{A}\,|v\rangle \otimes \mathbf{B}\,|w\rangle\,.$$

- It can be proved that $V \otimes W$ allows the following (natural) inner product $\langle \cdot, \cdot \rangle$:

$$\langle \sum_i a_i \,|v_i\rangle \otimes |w_i\rangle\,, \sum_j b_j \,|v'_j\rangle \otimes |w'_j\rangle\rangle = \sum_{i,j} a_i^* b_j \,\langle v_i|v'_j\rangle\,\langle w_i|w'_j\rangle\,.$$

3.1.2 Projectors and the Completeness Relation for Orthonormal Basis

A very important linear operators is projectors (or projections).

Definition 3.1.2: Projectors

A projector on a vector space V is a linear operator $\mathbf{P} : V \to V$ such that $\mathbf{P}^2 = \mathbf{P}$.

Geometrically, projectors represent the projection operation from V to its subspace (depend on P). Namely, if we have a vector $\mathbf{v} \in V$, $\mathbf{Pv}$ is a vector lies in the subspace of V that is defined according to $\mathbf{P}$; $\mathbf{P}^2 = \mathbf{P}$ just reflects the fact that once the vector $\mathbf{v}$ is brought to the subspace, further applications of $\mathbf{P}$ will not move it anymore.

Section 2.1.6 of [NC11] takes an alternative (and equivalent) way to define projectors. It considers a dimension-d vector space V and a dimension-k subspace $W \subseteq V$ (where $k \leq d$). By Gram-Schmidt procedure, it is easy to see that there is a set of orthonormal basis $\{|1\rangle\,, \ldots, |d\rangle\}$ such that $\{|1\rangle\,, \ldots, |k\rangle\}$ constitutes a set of orthonormal basis for W. Then the projector onto the subspace W can be defined as

$$\mathbf{P} = \sum_{i=1}^{k} |i\rangle\,\langle i|\,.$$

Then Definition 3.1.2 simply follows as a property. This approach also reveals the connection between projectors and *completeness relation* for orthonormal vectors.

to do

talk about the relation between *completeness relation* for orthonormal vectors and *projectors* to subspace. Useful materials can be found at Section 2.1.6 and Section 2.1.4 of [NC11].

An example: the space $\mathbb{R}^2$ is spanned by orthonormal basis $\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\}$. It is easy to check that it satisfies the completeness relation. However, when $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$ are treated as the orthonormal basis for the subspace $\mathbb{R}^2$ of a larger space $\mathbb{R}^3$, they should be augmented as $\{\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}\}$. In this form, they do not satisfies the completeness relation anymore. To fix that, we need to add the third element $\begin{pmatrix}0\\0\\1\end{pmatrix}$ in the set of orthonormal basis for $\mathbb{R}^3$.

3.1.3 Normal Operators and Spectral Decomposition

Definition 3.1.3: Normal Operator

A normal operator on a complex Hilbert space is a continuous linear operator $\mathbf{P}$ such that

$$\mathbf{P}^\dagger \mathbf{P} = \mathbf{P}\mathbf{P}^\dagger.$$

(Since the term "linear operators" can be used interchangeably with "matrices", people also refer to "normal operators" as "normal matrices".)

The following theorem provides an extremely important characterization of normal projectors. It is a special case of the famous Spectral Theorem (see also the discussion in Section 3.1.4).

Theorem 3.1.4: Unitary Diagonalization of Normal Matrices

An operator $\mathbf{A}$ on a complex Hilbert space is normal if and only if it is unitarily diagonalizable. Namely, it can be written as $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\dagger$, where $\mathbf{U}$ is a unitary matrix and $\mathbf{\Lambda}$ is a diagonal matrix.

Proof. By the Schur decomposition, we can write any complex matrix as $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\dagger$, where $\mathbf{U}$ is unitary and $\mathbf{\Lambda}$ is upper-triangular. This implies that if $\mathbf{A}$ is normal, we must have $\mathbf{\Lambda}\mathbf{\Lambda}^\dagger = \mathbf{\Lambda}^\dagger\mathbf{\Lambda}$ (i.e. $\mathbf{\Lambda}$ is also normal). Therefore, $\mathbf{\Lambda}$ must be diagonal: a normal upper triangular matrix is diagonal. The converse is obvious. ■

Theorem 3.1.4 appears in a slightly different form in [NC11, Section 2.1.6]. Since this form is more quantum-mechanics friendly, we present it in the following.

Theorem 3.1.5: Spectral Decomposition of Normal Operators

A linear operator $\mathbf{P}$ on a vector space V is normal if and only if it is diagonalizable with respect to some orthonormal basis for V, i.e. it can be decomposed as:

$$\mathbf{P} = \sum_i \lambda_i \left|i\right\rangle \left\langle i\right|, \tag{3.1}$$

where $(\lambda_i, \langle i|)$'s are the eigenvalue-eigenvector pairs of $\mathbf{P}$, and $\{|i\rangle\}_i$ form an orthonormal basis for V.

In terms of projectors, Equation (3.1) can also be written as $\mathbf{P} = \sum_i \lambda_i \mathbf{P}_i$, where λ_i are again the eigenvalues of $\mathbf{P}$, and $\mathbf{P}_i$ is the projector onto the λ_i eigenspace of $\mathbf{P}$. These projectors satisfy the completeness relation $\sum_i \mathbf{P}_i = I$, and the orthonormality relation $\mathbf{P}_i\mathbf{P}_j = \delta_{ij}\mathbf{P}_i$, where δ_{ij} is the Kronecker delta.

Here are some special normal operators (thus enjoying the spectral decomposition):

- **Unitary Operators:** operators represented by matrix U such that $U^\dagger U = I$.
- **Hermitian Operators:** operators represented by matrix H such that $U^\dagger = U$.

- **Positive Operators:** operators represented by positive semi-definite matrices. It can be proved that positive operators are necessarily hermitian. Note that density operators are necessarily positive (thus normal, thus diagonalizable).

3.1.4 Spectral Decomposition and Diagonalization in General

As we mentioned earlier, Theorem 3.1.5 is actually a part of the larger topic of spectral decomposition (aka eigendecomposition). We first need to define diagonalizable matrices (operators) formally.

Definition 3.1.6: Diagonalizable Matrices

A $n \times n$ matrix $\mathbf{A}$ over a field $\mathbb{F}$ is called diagonalizable (aka nondefective) if there exists an invertible matrix $\mathbf{P}$ such that $\mathbf{\Lambda} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$ is a diagonal matrix.

In the following, we give several equivalent characterizations of diagonalizable matrices. These claims can be proved easily from the definition of the matrix of an operator with respect to a basis.

- An $n \times n$ matrix $\mathbf{A}$ over a field $\mathbb{F}$ is diagonalizable if and only if the sum of the dimensions of its eigenspaces is equal to n, which is the case if and only if there exists a basis of $\mathbb{F}^n$ consisting of eigenvectors of $\mathbf{A}$. If such a basis $\{q_i\}_{i=1}^n$ has been found, then $\mathbf{A}$ can be factorized as $\mathbf{A} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1}$, where $\mathbf{P}$ is the $n \times n$ matrix whose i-th column is the q_i, and $\mathbf{\Lambda}$ is the diagonal matrix whose diagonal elements $\Lambda_{ii} = \lambda_i$. (The matrix $\mathbf{P}$ is known as a *modal matrix* for $\mathbf{A}$.)

- A linear map $\mathbf{T} : V \to V$ is diagonalizable if and only if the sum of the dimensions of its eigenspaces is equal to $\mathsf{dim}(V)$, which is the case if and only if there exists a basis of V consisting of eigenvectors of $\mathbf{T}$. With respect to such a basis, $\mathbf{T}$ will be represented by a diagonal matrix. The diagonal entries of this matrix are the eigenvalues of $\mathbf{T}$.

All the above can be summarized by the following lemma:

Lemma 3.1.7: Equivalence between Eigendecomposition and Diagonalization

A square matrix has eigendecomposition (aka spectral decomposition) if and only if it is diagonalizable.

With the above discussion, Theorem 3.1.5 (and Theorem 3.1.4) has a more natural interpretation: it just says that any $n \times n$ normal matrix A has exactly n orthonormal eigenverctors. In more details, if we put all the orthonormal eigenverctors column by column, they will form a $n \times n$ unitary matrix U; and $\mathbf{A}$ has the eigendecomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\dagger$. Note that $\mathbf{U}$ plays the role of $\mathbf{P}$ in Definition 3.1.6 as $\mathbf{U}^\dagger = \mathbf{U}^{-1}$.

3.2 The Four Postulates of Quantum Mechanics

From a mathematic point of view, the whole area of quantum computing can be derived from the following 4 postulates of quantum mechanics (together with linear algebra for Hilbert spaces). The formalism if taken verbatim from the amazing book by Nielsen and Chuang [NC11].

1. **(State.)** Associated to any isolated physical system is a complex vector space with inner product (aka Hilbert space) known as the state space of the system. The system is completely described by its density operator, which is a positive operator[1] ρ with trace one, acting on the state space of the system. If a quantum system is in the state ρ_i with probability p_i, then the density operator for the system is $\sum_i p_i \rho_i$.

2. **(Evolution.)** The evolution of a closed quantum system is described by a unitary transformation. That is, the state ρ of the system at time t_1 is related to the state ρ' of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2,
$$\rho' = U \rho U^\dagger.$$

3. **(Measurement.)** Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is ρ immediately before the measurement then the probability that result m occurs is given by
$$p(m) = \operatorname{tr}\Big(M_m^\dagger M_m \rho\Big),$$
and the state of the system after the measurement is
$$\frac{M_m \rho M_m^\dagger}{\operatorname{tr}\Big(M_m^\dagger M_m \rho\Big)}.$$
The measurement operators satisfy the completeness equation,
$$\sum_m M_m^\dagger M_m = I$$

4. **(Composition.)** The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state ρ_i, then the joint state of the total system is $\rho_1 \otimes \rho_2 \otimes \ldots \otimes \rho_n$.

3.3 Quantum Operations from a Mathematical Point of View

In this section, we present 3 equivalent interpretations of quantum operations (aka channels). A great explanation can be found here.

3.3.1 Stinespring's Representation of Quantum Channels

See this link for the origin of the name.

[1]This term "positive operator" comes from functional analysis, where positive operators are defined by positive semi-definite matrices.

As an implication of Schrödinger's equation, quantum operations can be classified as:

1. unitaries, which is captured by unitary operations on quantum states.
2. adding systems, which is captured by isometry operations (Definition 3.3.2) on quantum states.
3. partial trace.

All the above types of operations can be captured by a unitary process happening on a larger system (Hilbert space), followed by tracing out. This is sometimes referred to as "Church of Larger Hilbert Space". This is formally stated as Theorem 3.3.1.

Theorem 3.3.1: Stinespring Dilation Theorem

Let $T : S(H) \mapsto S(H)$ be a completely positive and trace-preserving (CTPT, see Section 3.3.3) map between states on a finite-dimensional Hilbert space H. Then there exists a Hilbert space K and a unitary operation U on $H \otimes K$ such that

$$T(\rho) = \mathrm{tr}_K \big(U(\rho \otimes |0\rangle\langle 0|)U^\dagger\big)$$

for all $\rho \in S(H)$, where tr_K denote the partial trace on the K-system.

Remark 3.3.1 (On Isometry)**.** The formal definition of isometry is presented in Definition 3.3.2. But in quantum computing, we usually focus on the special case where the metric spaces are Hilbert spaces. In the context, isometry means a map $V \in L(\mathbb{C}^n, \mathbb{C}^m)$ with $n \le m$ such that $\||\psi\rangle\|_1 = \|V|\psi\rangle\|_1$, or equivalently, $V^\dagger V = I_n$. It can be viewed as a generalization of unitary transformations between Hilbert spaces with different dimension.

Definition 3.3.2: Isometry

Let X and Y be metric spaces with metrics d_X and d_Y. A map $f : X \to Y$ is called an isometry or distance preserving if for any $a, b \in X$ one has: $d_Y\big(f(a), f(b)\big) = d_X(a, b)$.

3.3.2 Krause Operator Decomposition of Quantum Channels

This is also called operator-sum representation of quantum channels.

Theorem 3.3.3: Kraus Decomposition

Let $\mathcal{H}$ and $\mathcal{G}$ be Hilbert spaces of dimension n and m respectively, and Φ be a quantum operation between $\mathcal{H}$ and $\mathcal{G}$. Then, there are matrices $\{B_i\}_{1\ge i \ge nm}$ mapping $\mathcal{H}$ to $\mathcal{G}$ such that for any density matrix ρ,

$$\Phi(\rho) = \sum_i B_i \rho B_i^\dagger.$$

Conversely, any map Φ of this form is a quantum operation, provided that

$$\sum_i B^\dagger B_i \leq 1. \tag{3.2}$$

We remark that if the "$\leq$" sign is replaced with "$=$" in Inequality (3.2), Theorem 3.3.3 captures only trace-preserving operations; the above formalism (with the "$\leq$" sign) captures also non-trace-preserving operations. For more discussion, see [NC11, Section 8.2.3].

3.3.3 Axiomatic Definition of Quantum Operations (or the CTPT formalism)

The presentation of this form is taken from this lecture of MIT 8.371 course, which is equivalent to Theorem 3.3.3 when Inequality (3.2) takes "=" sign. It starts by asking "what properties should a general quantum operator satisfy?" The following conditions turn out to be complete:

1. Hermeticity Preserving: a hermitian input should lead to a hermitian output.

2. Trace Preserving: just as unitaries preserve length, our quantum operations should preserve trace.

3. Completely Positive: just like the non-negativity condition on stochastic maps. Positive means that if ρ is non-negative, then $\Phi(\rho)$ is non-negative. However, we need a stronger condition for it to be correct. We need to stipulate that if we act on any part of ρ it should stay positive. That is if ρ_{AR} is positive semi-definite, then $(\Phi \otimes I_R)(\rho)$ should also be positive semi-definite.

   (As a comparison, transpose is positive but not completely positive, as a side note: the partial positive transpose test (PPT) is one test of an entangled state: if PPT fails, it must be an entangled state)

We remark that [NC11, Section 8.2.4] contains a slightly different version of the above axiomatic representation, which is equivalent to Theorem 3.3.3 (with Inequality (3.2) as it is).

3.4 (In)distinguishability of Quantum States

In classical cryptography, the concept of distance is crucial for formal security proofs. According to the security goal, we may use different types of distances, e.g. statistical distance, computational distance (indistinguishability).

Therefore, a crucial step toward quantum information theory and cryptography would be to define a proper distance. One of the most useful definition is trace distance.

Definition 3.4.1: Trace Distance

The trace distance of two density matrices ρ and σ is defined as

$$T(\rho,\sigma) := \frac{1}{2}\operatorname{tr}|\rho-\sigma|,$$

where for a matrix A, $|A| = \sqrt{A^\dagger A}$.

On the Motivations for Trace Distance. Section 9.2 provides explanations on the choice of this concept as a useful measure for the distance among quantum states/systems. But I find that this lecture (with this video) approaches the concept in a more interesting way. It draws analogy between classical probability theory and quantum probability theory, and reveals the measure-theoretical reason behind the concept of trace distance by introducing Schatten norm and dual norm. This video by Prof. O'Donnell shares a similar perspective.

For quantum cryptography or information theory, this concept is useful mainly due to the following two properties (see [NC11, Section 9.2] for more details):

- $T(\rho,\sigma) = \max_P\{\operatorname{tr}\big(P(\sigma-\rho)\big)\}$, where the maximization may be taken alternately over all projectors P, or over all positive operators $P \le I$. Given the fact that POVM elements are positive operators that are $\le I$, this property implies that trace distance is equal to the difference in probabilities that a measurement outcome with POVM element P may occur, depending on whether the state is ρ or σ, maximized over all possible POVM elements P.

- **(Contractive.)** Suppose Ψ is a trace-preserving quantum operation. Let ρ and σ be density operators. Then $T\big(\Psi(\rho),\Psi(\sigma)\big) \le T(\rho,\sigma)$. This property says that (trace-preserving) quantum operations can never separate two quantum states farther than their original trace distance.

3.5 Supplementary Readings for Quantum Computation/Information/Crypt

- Intro to Quantum Computing (by Henry Yuen)
- Quantum Complexity (by Henry Yuen)
- The 11th BIU Winter School on Cryptography
- Interactive proofs with quantum devices (by Thomas Vidick).
- The Complexity of Quantum States and Transformations (by Scott Aaronson)

Chapter 4

Probability Theory

4.1 General Disjunction Rule of Events

Everyone is familiar with the following disjunction rule of two events:

$$\Pr[A_1 \vee A_2] = \Pr[A_1] + \Pr[A_2] - \Pr[A_1 \wedge A_2],$$

which can be straightforwardly demonstrated via Venn diagram.

In the following, we show the (less-familiar) extension of the above rule to n events.

$$\Pr[A_1 \vee \ldots \vee A_n] = \sum_{k=1}^{n} (-1)^{k-1} \sum_{\substack{\{i_1,\ldots,i_k\} \\ \in C_{k,n}}} \Pr[A_{i_1} \wedge \ldots \wedge A_{i_k}], \tag{4.1}$$

where $C_{k,n}$ is the set of all ordered k-uples $i_1 < \ldots < i_k$ of $[n]$.

Equation (4.1) can be easily proved by induction. But the writing could be painful, thus omitted. In the following we show the case for $n = 3$ to provide some intuition.

$$\Pr[A_1 \vee A_2 \vee A_3] = \sum_{i=1}^{3} \Pr[A_i] - \big(\Pr[A_1 \wedge A_2] + \Pr[A_1 \wedge A_3] + \Pr[A_2 \wedge A_3]\big) + \Pr[A_1 \wedge A_2 \wedge A_3]$$

4.2 Union Bound and the Probabilistic Method

(Put the definition/derivation here...)

An exemplary application of union bound and the probabilistic method is the proof of the following lemma, which is an important step in obtaining the famous Nisan-Wigderson PRG.

Lemma 4.2.1: Overlapping Subsets [NW94][1]

Let an (ℓ, k, d)-design be a set $\mathcal{I} = \{I_1, \ldots, I_m\}$, where each I_i is a size-k subset of $\{1, 2, \ldots, \ell\}$ such that any $|I_i \cap I_j| \leq d$ for all $i \neq j$.
If $\ell \geq 10k^2/d$, then there is an (ℓ, k, d)-design that achieves $m = 2^{d/10}$ and can be constructed in deterministic time $2^{O(\ell)}$.

[1]This formalization and proof are taken from [AB09].

Proof. On inputs ℓ, k, d with $\ell > 10k^2/d$, our algorithm A will construct an (ℓ, k, d)-design $\mathcal{I}$ with $2^{d/10}$ sets using the simple greedy strategy:

> Start with $\mathcal{I} = \emptyset$ and after constructing $\mathcal{I} = \{I_1, \dots, I_m\}$ for $m < 2^{d/10}$, search all subsets of $[\ell]$ and add to $\mathcal{I}$ the first k-sized set I satisfying the following condition (*): $|I \cap I_j| \leq d$ for every $j \in [m]$.

Clearly, A runs in $\mathsf{poly}(m)2^\ell = 2^{O(\ell)}$ time and so we only need to prove it never gets stuck. In other words, it suffices to show that if $\ell = 10n^2/d$ and $\{I_1, \dots, I_m\}$ is a collection of k-sized subsets of $[\ell]$ for $m < 2^{d/10}$, then there exists an k-sized subset $I \subseteq [\ell]$ satisfying (*). This can be shown by probability method. Namely, we do so by showing that if we pick I at random by choosing independently every element $x \in [\ell]$ to be in I with probability $2k/\ell$.

Since the expected size of I is $2k$, it follows from Chernoff bound that

$$\Pr[|I| \geq k] \leq 0.9. \tag{4.2}$$

Since the expected size of intersection $I \cap I_j$ is $2k^2/\ell < d/5$ for all $j \in [m]$, it follows again from Chernoff bound that

$$\forall j \in [m],\ \Pr[|I \cap I_j| \geq d] \leq 0.5 \cdot 2^{-d/10}.$$

Because $m \leq 2d/10$, the above inequality together with union bound implies:

$$\forall j \in [m],\ \Pr[|I \cap I_j| < d] \geq 1 - 0.5 \cdot 2^{-d/10} \cdot 2^{d/10} = 0.5. \tag{4.3}$$

Inequality (4.2) and Inequality (4.3) implies that with probability at least $0.9 \cdot 0.5 = 0.45$, the set I will simultaneously satisfy (*) and have size at least k. Since we can always remove elements from I without damaging (*), this completes the proof. ■

4.3 Averaging Argument

Consider the following simple fact: if the average of a set real numbers $\{a_i\}_{i \in [n]}$ is some c, then there must exist some $a_i \geq c$ (or $a_i \leq c$). This fact with some of its variants turns out to be very helpful in many cryptographic scenarios, especially in the security proof of protocols where non-uniform argument is used.

Before we present the most crypto-friendly version, see an example for how powerful this kind of argument can be (even) at its simplest form:

- Erdos argument for no-monochromatic-clique graph.

There are several interesting variants of this argument, see Appendix A.2.2 of [AB09]. In the following, we show a popular one that always appears when a proof wants to make use of the auxiliary (or random) tape of non-uniform Turing machines.

Lemma 4.3.1: Averaging Argument

If $X \in [0,1]$ and $E[X] = \mu$, then $\forall c < 1$ the following holds:

$$\Pr[X \leq c\mu] \leq \frac{1-\mu}{1-c\mu} \tag{4.4}$$

Remark 4.3.1. As one may already realized, the averaging argument shown in Lemma 4.3.1 has a similar flavor of the famous Markov Inequality (Lemma 4.5.1). Indeed, both of them say that a sampled random variable cannot differ too much from its expectation. Some papers refer to Markov inequality as the "averaging argument".

A Toy Example. An interesting application (of Lemma 4.3.1) that has some counter-intuitive implication: suppose you took a lot of exams, each with the score range $[1, 100]$. If your average score was 90, then in $\geq \frac{1}{2}$ fraction of these exams you scored ≥ 80.

The following corollary (of Lemma 4.3.1) is ubiquitous in the security proof of crypto protocols.

Corollary 4.3.2 (Averaging Argument)**.** If $a_1, a_2, \ldots, a_n$ are numbers in the interval $[0,1]$ whose average is ρ, then at least $\frac{\rho}{2}$ of the a_i's are at least as large as $\frac{\rho}{2}$. $\Diamond$

4.3.1 Exemplary Applications of the Averaging argument

Applications in Non-Uniform Argument. We demonstrate the usage of Corollary 4.3.2 by the following abstracted scenario. Consider a adversary Adv in some security reduction. Imagine that we want to build a machine $\mathcal{B}$ such that if Adv does something specific (w.l.o.g., say "outputting 1") with probability p^* (e.g. breaks the security property we are proving), $\mathcal{B}$ can make us of Adv to break some underlying assumptions with some probability polynomially-related to p^*. In this procedure, $\mathcal{B}$ may run Adv (internally) up to some stage and save the current state of Adv as st^*, which we usually say "freeze machine Adv at st^*" . Later, it may start Adv (directly) from st^* to finish the remaining steps, or to perform some specific operation (e.g. rewinding the steps after st^*).

A common task in this scenario is to estimate the probability that Adv outputs 1 when stating from st^*. For example, if rewinding is the concerned operation, this probability determines how many rewinds (the expected running time) are necessary for Adv to output 1 (again).

According to our assumption, we have $\Pr[\mathsf{Adv} = 1] = p^*$. But this probability is taken over all the randomness used by Adv, which might include the random tape of Adv, the distribution of the input and so on. Since we now freeze Adv at st^*, the probability of outputting 1 is not p^* anymore. We actually target the following conditional probability

$$\Pr[\mathsf{Adv} = 1 \mid \text{starting from } \mathsf{st}^*].$$

More formally, we should use S to denote the random variable that describe the possible status of Adv up to the "frozen point". We use Sup to denote the support of S . We are interested in the case when $\mathsf{S} = \mathsf{st}^*$. Let us consider the following decomposition:

$$\begin{aligned}\Pr[\mathsf{Adv} = 1] &= \sum_{\mathsf{st} \in \mathsf{Sup}} \Pr[\mathsf{Adv} = 1 \wedge \mathsf{S} = \mathsf{st}] \\ &= \sum_{\mathsf{st} \in \mathsf{Sup}} \Pr[\mathsf{Adv} = 1 \mid \mathsf{S} = \mathsf{st}] \cdot \Pr[\mathsf{S} = \mathsf{st}].\end{aligned}$$

The idea behind the proof of Corollary 4.3.2 tells us a useful fact that there are at least $p^*/2$ fraction of Sup such that from Adv resuming from these states will output 1 with probability at least $p^*/2$.[2] Using our notation, it guarantees the existence of a subset $\mathsf{Sup}' \subset \mathsf{Sup}$ such that the following holds:

(i) $|\mathsf{Sup}'| \geq \frac{p^*}{2}|\mathsf{Sup}|$, **and**

(ii) $\forall \mathsf{st} \in \mathsf{Sup}'$, $\Pr[\mathsf{Adv} = 1 \mid \mathsf{S} = \mathsf{st}] \geq \frac{p^*}{2}$.

In the common setting, p^* is usually noticeable. The above says that if $\mathcal{B}$ picks st uniformly at random, then with noticeable probability, the remaining part of Adv will finish outputting 1 (or satisfy some requirement) with noticeable probability. This usually suffices to finish the security reduction.

For a concrete example of the above approach, see the proof of soundness for the famous BJY protocol (a 4-round ZKAoK from any OWFs) [BJY97, Lemma 4.3].

Applications to "Truncated" Executions. In some scenarios, we need to do security reduction with an expected polynomial time adversary. This could be potentially problematic as the cryptographic assumptions are usually stated w.r.t. PPT adversaries. To address this problem, we can truncate the target adversary when it goes beyond some pre-fixed polynomial running time, and still hope to finish the reduction successfully with non-negligible probability.

Section 12.1 contains a detailed discussion (with concrete examples) about how to use averaging argument in such a "truncating" argument.

Applications in the Black-Box Separation Literature. Averaging argument is widely used in the proof of black-box separation results. In these applications, it usually appears in the following form (e.g., [HR04, CLMP12, Haj18]).

Let X and Y be random variables. Let $\mathsf{Event}_{X,Y}$ be some event depend on X and Y. Assume it is known that

$$\Pr_{X,Y}[\mathsf{Event}_{X,Y}] \geq 1 - \epsilon. \tag{4.5}$$

Then, the averaging argument[3] is used to show that for $c > 1$

$$\Pr_X\left[\Pr_Y[\mathsf{Event}_{X,Y}] \geq 1 - c \cdot \epsilon\right] \geq 1 - \frac{1}{c}, \tag{4.6}$$

[2]Note that I mean to say that following the same proof technique used to prove Corollary 4.3.2, we can prove the concerned fact. But this fact does not following immediately from Corollary 4.3.2 per se.

[3]As mentioned in Remark 4.3.1, "averaging argument" in this context usually refers to Markov Inequality

and also,

$$\Pr_X\left[\Pr_Y[\mathsf{Event}_{X,Y}] \geq 1 - \frac{1}{c}\right] \geq 1 - c\cdot\epsilon, \tag{4.7}$$

This procedure is so common that authors usually omit the details. Here, I provide a detailed derivation that may help a beginner to see the concrete math.

To do that formally, assume for contradiction that Inequality (4.6) does not hold. That is, there are a $\delta < 1 - \frac{1}{c}$ fraction of X such that $\Pr_Y[\mathsf{Event}_{X,Y} \mid X] \geq 1 - c\cdot\epsilon$. Call this δ fraction of X "good". Then, we have

$$\begin{aligned}
\Pr_{X,Y}[\mathsf{Event}_{X,Y}] &= \Pr_Y[\mathsf{Event}_{X,Y} \mid X \text{ is good}]\cdot\Pr_X[X \text{ is good}] + \Pr_Y[\mathsf{Event}_{X,Y} \mid X \text{ is bad}]\cdot\Pr_X[X \text{ is bad}]\\
&\leq 1\cdot\delta + (1 - c\cdot\epsilon)\cdot(1-\delta)\\
&= 1 - c\cdot\epsilon + c\cdot\epsilon\cdot\delta\\
&= 1 + c\cdot(\delta - 1)\epsilon\\
&< 1 - \epsilon \qquad\qquad (\text{since } \delta < 1 - 1/c).
\end{aligned}$$

This contradicts Inequality (4.5), thus finishing the proof.

If we replace the $\mathsf{Event}_{X,Y}$ with its negation, we will get the following version of averaging argument:

$$\Pr_{X,Y}[\mathsf{Event}_{X,Y}] \leq \epsilon \;\Rightarrow \begin{cases} \Pr_X\left[\Pr_Y[\mathsf{Event}_{X,Y}] \leq c\cdot\epsilon\right] \geq 1 - \dfrac{1}{c}\\ \Pr_X\left[\Pr_Y[\mathsf{Event}_{X,Y}] \leq \dfrac{1}{c}\right] \geq 1 - c\cdot\epsilon \end{cases}. \tag{4.8}$$

<u>Another Proof.</u> There is another way to derive Inequality (4.6) from Inequality (4.5), using Markov's inequality (Lemma 4.5.1). First, observe that

$$\Pr_{X,Y}[\mathsf{Event}_{X,Y}] = \mathop{\mathbb{E}}_X\left[\Pr_Y[\mathsf{Event}_{X,Y}]\right] \quad \left(\text{and } \Pr_{X,Y}[\neg\mathsf{Event}_{X,Y}] = \mathop{\mathbb{E}}_X\left[\Pr_Y[\neg\mathsf{Event}_{X,Y}]\right]\right).$$

Thus, it follows from Inequality (4.5) that

$$\begin{aligned}
&\mathop{\mathbb{E}}_X\left[\Pr_Y[\neg\mathsf{Event}_{X,Y}]\right] \leq \epsilon\\
\Rightarrow &\Pr_X\left[\Pr_Y[\neg\mathsf{Event}_{X,Y}] \geq c\cdot\epsilon\right] \leq \frac{1}{c} \qquad (\text{by Markov's Inequaltiy as in Lemma 4.5.1})\\
\Rightarrow &\Pr_X\left[\Pr_Y[\neg\mathsf{Event}_{X,Y}] \leq c\cdot\epsilon\right] \geq 1 - \frac{1}{c}\\
\Rightarrow &\Pr_X\left[\Pr_Y[\mathsf{Event}_{X,Y}] \geq 1 - c\cdot\epsilon\right] \geq 1 - \frac{1}{c},
\end{aligned}$$

where the last inequality is exactly Inequality (4.6).

4.4 Berry-Esseen Theorem

To do...

Check these resources:

- Ryan O'donnell's lecture
- This Wikipedia page.

Recently, Tomaszewski's Conjecture was resolved [KK20]. Berry-Esseen inequality plays an important role in the final proof.

4.5 Concentration Bounds

4.5.1 Markov Inequality

The common form of Markov Inequality is shown below:

Lemma 4.5.1: Markov Inequality

If X is a nonnegative random variable and $a > 0$, then the probability that X is at least a is at most the expectation of X divided by a, i.e.,

$$\Pr[X \geq a] \leq \frac{E(x)}{a}.$$

Let $a = c \cdot E(X)$ (where $c > 0$); then, we can rewrite the previous inequality as:

$$\Pr[X \geq c \cdot E(X)] \leq \frac{1}{c}.$$

The following version is Markov inequality is useful if one want to lower-bound the value of a random variable. It can be proved by applying Markov's inequality (Lemma 4.5.1) to $X = B - Y$. Note that X is a non-negative random variable as $Y < B$.

Corollary 4.5.2: The Reverse Markov Inequality

Let Y be a random variable that is never larger than $B \in \mathbb{R}$. Then, for all $a < B$,

$$\Pr[Y \leq a] \leq \frac{B - E(Y)}{B - a}.$$

The following is a widely used argument in cryptography. It is so standard that many authors refer to it without a proof. In [DGH+19], the authors formalize it under the name "Markov Inequality for Advantages"[4]. The reason why it is called "Markov Inequality" remains mysterious to me. Maybe it is because the proof and the intuition behind this bound goes in the same sense as the standard Markov Inequality?

[4]This is the first place where I saw such a formalism But it is possible that it already appeared somewhere else.

Theorem 4.5.1 (Markov Inequality for Advantages)**.** Let $A(Z)$ and $B(Z)$ be two random variables depending on a random variable Z and potentially additional random choices. Assume that

$$\left|\Pr_Z[A(Z)=1]-\Pr_Z[B(Z)=1]\right| \geq \varepsilon \geq 0.$$

Then

$$\Pr_Z\left[\left|\Pr[A(Z)=1]-\Pr[B(Z)=1]\right| \geq \frac{\varepsilon}{2}\right] \geq \frac{\varepsilon}{2}.$$

$\Diamond$

Proof. The idea is to condition the event on $\left|\Pr[A(Z)=1]-\Pr[B(Z)=1]\right| \geq \frac{\varepsilon}{2}$. Let

$$a = \Pr_Z\left[\left|\Pr[A(Z)=1]-\Pr[B(Z)=1]\right| \geq \frac{\varepsilon}{2}\right].$$

We have $\varepsilon \leq a \times 1 + (1-a) \times \frac{\varepsilon}{2}$. Since $0 \leq 1-a \leq 1$, we obtain $\varepsilon \leq a + \frac{\varepsilon}{2}$. The inequality now follows. ■

4.5.2 Chebyshev Inequality

4.5.3 Chernoff Bound

Another version

Wikipedia has an exhaustive explanation for this topic.
The lecture of Prof. O'donnell also gives a great presentation for Chernoff Bounds

Theorem 4.5.2 (Chernoff Bound)**.** Let X_i be i.i.d. random variables such that $0 \leq X_i \leq 1$. Let $\mu = \mathbb{E}[\sum_i X]$. For and $\varepsilon > 0$, we have

- $\Pr[X \leq (1-\delta)\cdot\mu] \leq \exp\left(-\frac{\delta^2\mu}{2}\right)$
- $\Pr[X \geq (1+\delta)\cdot\mu] \leq \exp\left(-\frac{\delta^2\mu}{2+\delta}\right)$

$\Diamond$

Interestingly, if we know that $L \leq \mu \leq H$, the following bounds hold:

- $\Pr[X \leq (1-\delta)\cdot L] \leq \exp\left(-\frac{\delta^2 L}{2}\right)$
- $\Pr[X \geq (1+\delta)\cdot H] \leq \exp\left(-\frac{\delta^2 H}{2+\delta}\right)$

The following corollary of Chernoff bound is taken from Prof. O'donnell's notes for his lecture on Chernoff Bound. The poof was left as an exercise.

Lemma 4.5.3 (Sampling Lemma)**.** Let μ be the unknown mean for a random variable $0 \leq X \leq 1$. Let $x_1, \ldots, x_n$ be n independent samples of X. Let $\hat{\mu}$ be the empirical mean of x_i's, i.e. $\hat{\mu} := \frac{x_1+\cdots+x_n}{n}$. Then for any $0 < \varepsilon, \delta < 1$ such that $n \geq \frac{3\ln(1/\delta)}{\varepsilon^2}$, the following holds:

$$\Pr\left[|\hat{\mu}-\mu| \leq \varepsilon\right] \geq 1-\delta.$$

◊

Theorem 4.5.4 (Chernoff Bound)**.** (Put the general form here ....) ◊

Chernoff Bounds from this MIT lecture notes

Theorem 4.5.5 (Chernoff Bound (Upper Tail))**.** Let $X = \sum_{i=1}^{n} X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i's are independent. Let $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i$. Then the following holds for any $0 < \delta < 1$:

$$\Pr[X \geq (1+\delta) \cdot \mu] \leq \exp\big(-\frac{\delta^2 \mu}{2+\delta}\big)$$

◊

Probably the most widely used form of Chernoff bound is the following one:

Corollary 4.5.6. Let $X_1, \ldots, X_n$ be independent variables with $0 \leq X_i \leq 1$ for all $1 \leq i \leq n$, denote $\mu = \mathbb{E}[\frac{\sum_{i=1}^{n} X_i}{n}]$. Then, for any $\varepsilon > 0$,

$$\Pr\big[\big|\frac{\sum_{i=1}^{n} X_i}{n} - \mu\big| \geq \varepsilon\big] \leq 2^{-\varepsilon^2 \cdot n} \tag{4.9}$$

◊

The take-away from Chernoff bound is very simple: The empirical mean of a bunch of independent random variables is approaching the expectation (of the mean) in an exponentially fast manner.

4.5.4 An Exemplary Application of Concentrations Bounds: the Goldreich-Levin Theorem

(Todo: the proof of Goldreich-Levin theorem serves as an beautiful example where Markov's inequatliy, Chebyshev's inequality, and Chernoff bound are used. So, now is a prefect time to present this famous theorem. Use the version from Luca's lecture notes.)

4.5.5 Hoeffding Inequality

Theorem 4.5.7 (Hoeffding's Inequality [Hoe63])**.** Let $X_1, \ldots, X_n$ be independent variables with $b_i \leq X_i \leq a_i$ for all $1 \leq i \leq n$, denote $\mu = \mathbb{E}[\frac{\sum_{i=1}^{n} X_i}{n}]$. Then, for any $\varepsilon > 0$, we have:

$$\Pr\big[\big|\frac{\sum_{i=1}^{n} X_i}{n} - \mu\big| \geq \varepsilon\big] \leq 2 \cdot e^{-\frac{2 \cdot \varepsilon^2 \cdot n^2}{\sum_{i=1}^{n}(b_i - a_i)^2}} \tag{4.10}$$

The following single-side form also holds:

$$\Pr\big[\frac{\sum_{i=1}^{n} X_i}{n} - \mu \geq \varepsilon\big] \leq e^{-\frac{2 \cdot \varepsilon^2 \cdot n^2}{\sum_{i=1}^{n}(b_i - a_i)^2}} \tag{4.11}$$

$\Diamond$

4.6 Stochastic Process

4.6.1 Doob's Martingale

4.7 Coupon-Collection Problems

Lemma 4.7.1. Suppose that there are m different types of coupons, and each time one obtains a coupon of type i $(1 \le i \le m)$ with probability $\frac{1}{n}$, where $n \ge m$ is a parameter. Note that with probability $1 - \frac{m}{n}$, one does not obtain any coupon (or obtains an "empty coupon"). Then the expected number of coupons one need amass before obtaining k $(1 \le k \le m)$ different types of non-empty coupons is

$$n \cdot (H_m - H_{m-k}) \tag{4.12}$$

where $H_t \coloneqq 1 + \frac{1}{2} + \ldots + \frac{1}{t}$ is the t-th harmonic number for $t \in \mathbb{N}$. $\Diamond$

Proof. Let $X(k)$ denote the number of coupons collected before k different types of coupons is attained. We need to compute $E[X(k)]$. we define X_j $(j = 0, 1, ..., k-1)$ to be the random variable representing the number of additional coupons that need be obtained after j distinct types have been collected in order to obtain another distinct type, and we note that

$$X(k) = X_0 + X_1 + \ldots + X_{k-1}.$$

When j distinct types of coupons have already been collected, a new coupon obtained will be of a distinct type with probability $(m-j)/n$. Therefore

$$\Pr[X_j = k] = \frac{m-j}{n}\left(\frac{n-m+j}{n}\right)^{k-1} \quad k \ge 1$$

or, in other words, X_j is a geometric random variable with parameter $(m-j)/n$. Hence, $\mathbb{E}[X_j] = \frac{n}{m-j}$ implying that

$$\begin{aligned}\mathbb{E}[X(k)] &= \frac{n}{m-k+1} + \frac{n}{m-k+2} + \ldots + \frac{n}{m-1} + \frac{n}{m} \\ &= n(H_m - H_{m-k})\end{aligned}$$

where H_t is the t-th harmonic number for $t \in \mathbb{N}$. ■

(**An application for knowledge Extractors.** We can use the above lemma in the following way: in our setting, n is the total nubmer of challenges, m is the set of "good" challenges (i.e. the prover will answer), k is the number of distinct challenges needed to extract a valid witness. If k is a polynomial and m is a super-polynomial on security parameter λ, we can assume that $m - k + 1 \ge m/2$. Then the expected running time of the knowledge extractor can be bounded as

$$\begin{aligned}\mathbb{E}[X(k)] &= \frac{n}{m-k+1} + \frac{n}{m-k+2} + \ldots + \frac{n}{m-1} + \frac{n}{m} \\ &\le n\frac{k}{m-k+1} = k \cdot \frac{n}{m-k+1} \le k \cdot \frac{n}{2m} = 2k\frac{n}{m} = \mathsf{poly}(\lambda)\end{aligned}$$

Since both k and $\frac{n}{m}$ are polynomials of λ, the expected running time $\mathbb{E}[X(k)]$ is also upper-bounded by a polynomial of λ.

)

Chapter 5

Number Theory

5.1 Prime Number Distribution

A useful link
Gauss's bound
Chebyshev bound
Bertrand's postulate
Talk about the relation between prime number distribution and Reimann Hypothesis

5.2 Euler's Totient Function

Euler's theorem
(Note that if $N = p \cdot q$, where p and q are two primes, then once we know $\phi(N)$, it is easy to factor N. To do that,

1. Note that

$$\begin{aligned} &\phi(N) = (p-1)(q-1) = N - (p+q) + 1 \\ \Rightarrow &p + q = N + 1 - \phi(N) \end{aligned} \tag{5.1}$$

2. p and q can be easily solved from Equ. 5.1 and $N = p \cdot q$.

In summary, computing $\phi(N)$ is equivalent to factorizing N when N is the product of two primes. (The other direction is trivial, i.e. it is easy to compute $\phi(N)$ given the factorization of N.)

)

repeated squaring
Fermat's little theorem
Chinese remainder theorem

5.3 Quadratic Residues

Lemma 5.3.1 (text)**.** Let $p > 2$ be prime. Every quadratic residue in $\mathbb{Z}_p^*$ has exactly two square roots. ◊

Th Quadratic Residuosity (QR) assumption was originally formalized in [GM84], to construct the well-known Goldwasser-Micali PKE scheme. Another very simple and interesting application is given by Kushilevitz and Ostrovsky [KO97], where they build the first computational Private Information Retrieval protocol in the single database setting with sub-linear communication complexity. More specifically, they achieve communication complexity $O(n^{\varepsilon})$ for any $\varepsilon > 0$, where n is the size of the database.

(There are two constructions in [KO97]. They start with the first construction which achieves communication complexity $O(n^{0.5+\varepsilon})$. Based on the first construction, they build their final scheme which achieves communication complexity $O(n^{\varepsilon})$. But the first construction is very simple. It can be presented here as a good demonstration of the power of QR assumption.)

5.4 Composite Residues

This assumption gives the well-know Paillier [Pai99] and Dåmgard-Jurik [DJ01] cryptosystems. This assumption relies on the group $\mathbb{Z}^*_{N^2}$, where N is the product of two equal-length primes. The following theorem summarize important properties of $\mathbb{Z}^*_{N^2}$ to our interest.

Theorem 5.4.1. Let $N = pq$, where p, q are distinct odd primes of equal length. Then:

1. $\mathsf{gcd}(N, \phi(N)) = 1$.
2. For any integer $a \geq 0$, we have $(1+N)^a = (1+aN) \bmod N^2$.

   As a consequence, the order of $(1+N)$ in $\mathbb{Z}^*_{N^2}$ is N.
3. $\mathbb{Z}_N \times \mathbb{Z}^*_N$ is isomorphic to $\mathbb{Z}^*_{N^2}$, with isomorphism $f : \mathbb{Z}_N \times \mathbb{Z}^*_N \to \mathbb{Z}^*_{N^2}$ given by

$$f(a, b) = (1+N)^a \cdot b^N \bmod N^2$$

   where the operation in $\mathbb{Z}_N \times \mathbb{Z}^*_N$ is defined as $(a_1, b_1) \cdot (a_2, b_2) = (a_1 + a_2, b_1 \cdot b_2)$.

$\Diamond$

Define the subset of N-th residues in $\mathbb{Z}^*_{N^2}$ as $\mathsf{Res}(N^2)$, using Theorem 5.4.1, we can show that very element in $\mathsf{Res}N^2$ is of the form (a, b) if written in the isomorphic group $\mathbb{Z}_N \times \mathbb{Z}^*_N$. Moreover, this characterization is sufficient. We summarize this in the following corollary.

Corollary 5.4.2. Let $N = pq$, where p, q are distinct odd primes of equal length. Denote the set of N-th residues modulo N^2 by $\mathsf{Res}(N^2)$. Then:

$$\mathsf{Res}(N^2) < \mathbb{Z}^*_{N^2} \quad \text{and} \quad \mathsf{Res}(N^2) \cong \{(0, b) \mid b \in \mathbb{Z}^*_N\} < \mathbb{Z}_N \times \mathbb{Z}^*_N$$

$\Diamond$

We are now ready to present the decisional composite residuosity (DCR) assumption. Intuitively, this assumption conjecctures that it is infeasible to distinguish a uniform element of $Z^*_{N^2}$ from a uniform element of $\mathsf{Res}(N^2)$. Formally,

Assumption 5.4.3 (DCR Assumption)**.** let $\mathsf{GenModulus}$ be a polynomial-time algorithm that, on input 1^λ, outputs (N, p, q) where $N = pq$, and p and q are λ-bit primes. The DCR assumption is that there exist a $\mathsf{GenModulus}$ algorithm such that for any PPT adversary Adv, the following holds:

$$\big|\Pr[\mathsf{Adv}(N, a) = 1] - \Pr[\mathsf{Adv}(N, b) = 1]\big| \leq \mathsf{negl}(\lambda)$$

where $a \xleftarrow{\$} \mathsf{Res}(N^2)$ and $b \xleftarrow{\$} \mathbb{Z}^*_{N^2}$.

5.5 Chinese Remainder Theorem

Chapter 6

Hash Functions

useful links for this chapter:

- CMU Algorithms in the Real World course

(Here (or may be at the end of this chapter, discuss about the difference and relation between IT-secure hashing and cryptographic hashing.))

The following paragraph is quoted from [HL18], which gives many examples for the application of cryptographic hashing:

- *Cryptographically secure hash functions are a fundamental building block in cryptography. Some of their most ubiquitous applications include the construction of digital signature schemes [NY89], efficient CCA-secure encryption [BR93], succinct delegation of computation [Kil94], and removing interaction from protocols [FS87]. In their most general form, hash functions can be modeled as "random oracles" [BR93], in which case it is heuristically assumed that an explicitly described hash function H (possibly sampled at random from a family) behaves like a random function, as far as a computationally bounded adversary can tell.*

6.1 Collision Resistant Hash Family

Collision Resistant Hash Functions, usually denoted as CRHF, was first formalized explicitly by Damgård [Dam88].

(to be done ...)

Definition 6.1.1 (Collision Resistant Hash Family). (to be done ...)

$\Diamond$

6.1.1 Merkle Hashing Trees and the Extraction Lemma

The following formalism of Merkle hashing trees is taken from [HHPS11].

Denote by $MT_{h,n}(X)$ the binary Merkle tree over string X using n-bit leaves and the hash function $h : \{0,1\}^{2n} \leftarrow \{0,1\}^n$. For each node k in the tree $n \in MT_{h,n}(X)$, we denote by v_k the value associated with that node. That is, the value of a leaf is the corresponding block of X, and the value of an intermediate node $n \in MT_{h,n}(X)$ is the hash $v_k = h(v_\ell \| v_r)$ where v_ℓ, v_r are the values for the left and right child of k, respectively. ((If one of the children of a node is missing from the tree then we consider its value to be the empty string.))

For a leaf node $x \in MT_{h,n}(X)$, the sibling path of x consists of the value v_x and also the values of all the siblings of nodes on the path from x to the root. Given the index of a leaf $x \in MT_{h,n}(X)$ and a sibling path for x, we can compute the values of all the leaves on the x-to-root path itself in a bottom-up fashion by starting from the two leaves and then repeatedly computing the value of a parent as the hash of the two children values.

We say that an alleged sibling path $P = (v_x, v_{k_0}, v_{k_1}, \ldots, v_{k_i})$ is valid with respect to $MT_{h,n}(X)$ if i is indeed the height of the tree and the root value as computed on the sibling path agrees with the root value of $MT_{h,n}(X)$. Note that in order to verify that a given alleged sibling path is valid, it is sufficient to know the number of leaves and the root value of $MT_{h,n}(X)$. We also note that any two different valid sibling paths with respect to the same Merkle tree imply in particular a collision in the hash function.

Merkle Tree Proof Protocol and an Extraction Lemma. Merkle trees play an important role in interactive arguments (i.e. computationally sound proofs). It can be used in the scenario where an PPT prover P wants to hash a long string and later proves to a verifier V that the hashing is honestly done w.r.t. some preimage string, without disclosing the string in full. We present this protocol (due to [Kil92]) in Protocol 6.1.1.

Protocol 6.1.1: Merkle Tree Hash-and-Prove

Let $\mathcal{H}$ be a collision-resistant hash family. The protocol where the prover hash-and-proves w.r.t. a string X proceeds in the following way:

1. The verifier V samples a function $h \overset{\$}{\leftarrow} \mathcal{H}$ and sends it to the prover.
2. The prover P builds Merkle tree $MT_{h,n}(X)$ using h received from V. It then sends the root value v and the number of leaves s to V.
3. V samples uniformly at random u distinct numbers $(p_1, \ldots, p_u) \in [s]^u$, indicating the leaves it wants to verify. V sends these values to S.
4. The prover replies with these leaves specified by $(p_1, \ldots, p_u)$ and with a sibling path for each one of them, and the verifier accepts if all these sibling paths are valid.

The soundness of Protocol 6.1.1 is captured by the following lemma, which basically says that any PPT prover P^* that manages to convince the verifier with good probability must know (in the sense of *argument of knowledge*) the preimage string.

Lemma 6.1.2: Merkle Tree Extraction [HHPS11]

There exists a black-box extractor K with oracle access to a Merkle-tree prover that has the following properties:

- For every prover P and $v \in \{0,1\}^*$, $s, u \in \mathbb{N}$, and $\delta \in [0,1]$, $K^P(v, s, u, \delta)$ makes at most $u^2 s(\log(s) + 1)/\delta$ calls to its prover oracle P;
- Fix any hash function h and input string X with s leaves of n-bits each, and let v be the root value of $MT_{h,n}(X)$. Also fix some $u \in \mathbb{N}$ and a prover's remaining strategy $P^* = P^*(h, X, u)$ for Step 3 and Step 4 (that may depend on h, X and u). Then if P^* has probability at least $(1 - \alpha)^u + \delta$ of convincing the verifier in the Merkle-tree protocol $MTP_h(v, s, u)$ (for some $\alpha, \delta \in (0, 1]$), then with probability at least $1/4$ (over its internal randomness) the extractor

$K^{P^*}(v, s, u, \delta)$ outputs values for at least a $(1-\alpha)$-fraction of the leaves of the tree, together with valid sibling paths for all these leaves.

Note that the proof of Lemma 6.1.2 does not rely on collision resistance of the hash function, it is merely a information-theoretical result. But it is usually used in conjunction with the collision-resistance property of hash functions to establish cryptographic results such as computational soundness or argument of knowledge property.

6.2 Universal One-way Hash Family

Another useful cryptographic (thus based on hardness assumptions) hashing is the *universal one-way hashing*. It was proposed in [NY89]. ((More discussion and applications can be found there.)). Roughly speaking, the definition starts with Adv picking a input x_1 before it learns the function. Then we sample a function from the family and give it to Adv. The goal of Adv is to find a second input x_2 which shares the same image as that of x_1, under the sampled hash function.

Definition 6.2.1 (Universal One-Way Hash Family)**.** (to be done ...)

◊

6.3 Universal Hash Family

This notion of universal hashing, which bounds the collision probability of a hash function in a statistical sense, dates back to [CW79, WC81].

Definition 6.3.1 (Universal Hash Family)**.** A family $\mathcal{H} = \{h_k\}_k$ of hash functions from domain $\mathcal{D}$ to range $\mathcal{R}$ is *universal* if $\forall x_1 \neq x_2 \in \mathcal{D}$,

$$\Pr[h_k \xleftarrow{\$} \mathcal{H} : h_k(x_1) = h_k(x_2)] \leq \frac{1}{|\mathcal{R}|} \tag{6.1}$$

◊

A simple example of universal hash family from $\mathcal{D} = \{0,1\}^k$ to $\mathcal{R} = \{0,1\}^n$ is

$$h_A(x) = A \cdot x$$

where $A \xleftarrow{\$} \{0,1\}^{n \cdot k}$ is interpreted as a $n \times k$ matrix and x is interpreted as a $k \times 1$ vector. The calculations are done modulo 2.

6.4 Pair-wise Independent Hash Family

Definition 6.4.1 (Pair-wise Independent Hash Family)**.** A family of hash functions $\mathcal{H}$ is *pairwise independent* if $\forall x_1 \neq x_2 \in \mathcal{D}$ and $\forall y_1, y_2 \in \mathcal{R}$,

$$\Pr[h_k \xleftarrow{\$} \mathcal{H} : h_k(x_1) = y_1 \wedge h_k(x_2) = y_2] = \frac{1}{|\mathcal{R}|^2} \tag{6.2}$$

$\Diamond$

Note that in equation (6.1) for universal hash family, the probability is bounded by "$\leq$". But in the equation (6.2), the symbol is "$=$". Actually, some authors also use "$\leq$" when defining pair-wise hash family. It does not matter that much since, in applications, it usually suffices the purpose once the collision probability is $\frac{1}{|\mathcal{R}|}$. I guess the tradition of using "$=$" is the following: the concept of pair-wise independent hashing is analogous to the concept of independence in probability theory, i.e. $\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B]$, where "$=$" symbol is used.

(I haven't checked whether there exist an construction that achieves probability strictly smaller than $\frac{1}{|\mathcal{R}|}$)

(Give an exmple of pair-wise independent hashing. e.g. $h_{a,b}(x) = ax + b$)

Pair-wise independence can be generalized to the following concept of *k-wise independence*.

Definition 6.4.2 (k-wise Independent Hash Family)**.** A family of hash functions $\mathcal{H} = \{h_i\}_i$ is *k-wise independent* if $\forall x_1 \neq \ldots \neq x_k \in \mathcal{D}$ and $\forall y_1, \ldots, y_k \in \mathcal{R}$,

$$\Pr[h_k \xleftarrow{\$} \mathcal{H} : h_i(x_1) = y_1 \wedge \ldots \wedge h_i(x_k) = y_k] = \frac{1}{|\mathcal{R}|^k} \tag{6.3}$$

$\Diamond$

Here are some obvious facts about k-wise independent hash family

Fact 6.4.3. Suppose $\mathcal{H}$ is a k-wise independent hash family for $k \geq 2$. Then

1. $\mathcal{H}$ is also $(k-1)$-wise independent.
2. For any $x \in \mathcal{D}$ and $y \in \mathcal{R}$, $\Pr[h \xleftarrow{\$} \mathcal{H} : h_i(x) = y] = \frac{1}{|\mathcal{R}|}$.
3. $\mathcal{H}$ is universal.

Remark 6.4.4 (On the ambiguous usage of "2-universal")**.** Usually, k-wise independent hash family is also called "k-universal" hash family [WC81], and the one given in Definition 6.3.1 is called "universal". But there are a few authors referring to Definition 6.3.1 as "2-universal", namely "universal" and "2-universal" are simply different names for the same property to them. In addition, some researchers refer to Definition 6.3.1 as "weakly 2-universal" and they refer to "pair-wise independent" as "strongly 2-universal". And when they say "2-universal", they by default mean "weakly 2-universal", i.e. Definition 6.3.1. One of such authors is Vadhan [Vad12].

6.5 Bloom Filter

Chapter 7

Pseudorandomness

7.1 Leftover Hash Lemma

In this section, we play with one of the most important lemma – Leftover Hash Lemma (LHL). Introduced first in [ILL89], it has since found numerous applications in the realms of complexity theory/quantum computing/(randomized) algorithm/information theory/cryptography. To give a few examples from cryptography, LHL was used to build Leakage-Resilient Encryption [HLWW13], Deterministic Encryption [BFO08], Fully Homomorphic Encryption [Gen09] and Program Obfuscation [BLMZ18] etc.

Roughly, LHL says that a universal hash function constitutes a good randomness extractor, "smoothing out" an input distribution to nearly uniform on its range, provided that the former has sufficient min-entropy. LHL can be generalized to the average conditional min-entropy setting [DORS03].

Definition 7.1.1 (Statistical Distance)**.** Let X and Y be two random variables with range U. Then the statistical distance between X and Y is defined as

$$\Delta(X,Y) = \frac{1}{2}\sum_{u\in U}\big|\Pr[X=u]-\Pr[Y=u]\big| \tag{7.1}$$

For $\varepsilon \geq 0$, we also define the notion of two distributions being ε-close:

$$X \approx_\varepsilon Y \Leftrightarrow \Delta(X,Y) \leq \varepsilon.$$

◊

Definition 7.1.2 (Min-Entropy)**.** The min-entropy $\mathsf{H}_\infty(X)$ of a random variable X is defined as

$$\mathsf{H}_\infty(X) = -\log\Big(\max_x\big\{\Pr[X=x]\big\}\Big) = \min_x\Big\{-\log\big(\Pr[X=x]\big)\Big\}. \tag{7.2}$$

If $\mathsf{H}_\infty(X) \geq k$, we call X a k-source.

◊

Lemma 7.1.3 (Leftover Hash Lemma [ILL89])**.** Let $\mathcal{H} = \{h_i\}_{i\in\mathcal{I}}$ be a universal hash family[1] $\mathcal{I}\times\mathcal{D}\to\mathcal{R}$ with $|\mathcal{D}| = 2^n$ $|\mathcal{R}| = 2^\ell$ for some $n,\ell > 0$. Let $\mathsf{Ext}(x,i) = h_i(x)$. For any random variable X on support $\mathcal{D}$, the following holds:

$$\Delta\Big(\big(\mathsf{Ext}(X,U_\mathcal{I}),U_\mathcal{I}\big),\big(U_\mathcal{R},U_\mathcal{I}\big)\Big) \leq 2^{-\left(\frac{\mathsf{H}_\infty(X)-\ell}{2}+1\right)} \tag{7.3}$$

[1]For the definition of universal hash family, refer to Def. 6.3.1.

or equivalently (but easier to interpret),

$$\ell \leq \mathsf{H}_\infty(X) - c \quad \Rightarrow \quad \Delta\Big(\big(\mathsf{Ext}(X, U_\mathcal{I}), U_\mathcal{I}\big), \big(U_\mathcal{R}, U_\mathcal{I}\big)\Big) \leq 2^{-(\frac{c}{2}+1)} \tag{7.4}$$

where $U_\mathcal{I}$ and $U_\mathcal{R}$ are uniform distributions on $\mathcal{I}$ and $\mathcal{R}$ respectively.

In particular, to achieve statistical distance ε, we need to set

$$\ell \leq \mathsf{H}_\infty(X) - 2\log\big(\frac{1}{\varepsilon}\big) + 2$$

◊

Proof. A simple but elegant proof is given in Reyzin's lecture notes. ■

Remark 7.1.4. It seems different people formalize LHL in different way. Reyzin's version (link) assumes universal hashing, but Rubinfiel's version (link) assumes 2-universal hashing. Also, [PW08] also assumes pairwise independent hash function. (Is it true that if we assume pairwise indenpedent hash function, then we will only need $\ell \leq \mathsf{H}_\infty(x) - 2\log\big(\frac{1}{\varepsilon}\big)$?)

How to Explain LHL to a Kid. Typically, we use hash functions for compression. Smaller range of a hash family (thus more compression achieved) means more information loss on the input. LHL takes advantage of this property to build *randomness extractors* (see Section 7.2) from universal hash families in the following way: even if the input distribution has low entropy, by hashing it with a *uniformly chosen* member from a universal hash family with proper compression rate, we can always "smooth" it to an (almost) uniform output. Parametrically, for an input with min-entropy k, if we compress it to c bits shorter than k, i.e. the output has length $m = k - c$, the joint distribution of the output and the hash key will $(\frac{1}{2})^{\frac{c}{2}+1}$-close to uniform distribution. Thus, the statical distance is exponentially small on the amount compressed below the min-entropy.

(Also, talk about the average-min entropy and the extended LHL. Check [BFO08], Reyzin's lecture notes and Yu Yu's lecture notes.)

Definition 7.1.5 (Conditional Min-Entropy and Average Min-Entropy)**.** Let A, B be random variables. The conditional min-entropy $\mathsf{H}_\infty(A \mid B = b)$ is defined as

$$\mathsf{H}_\infty(A \mid B = b) = -\log\Big(\max_a \Big\{ \Pr[A = a \mid B = b] \Big\}\Big) = \min_a \Big\{ -\log\Big(\Pr[X = a \mid B = b]\Big)\Big\}. \tag{7.5}$$

The average min-entropy $\widetilde{\mathsf{H}}_\infty(A \mid B)$ is defined as

$$\widetilde{\mathsf{H}}_\infty(A \mid B) = -\log\Big(\mathbb{E}_B\Big[\max_a\{\Pr[A = a \mid B]\}\Big]\Big) = -\log\Big(\mathbb{E}_B\Big[2^{-\mathsf{H}_\infty(A \mid B=b)}\Big]\Big). \tag{7.6}$$

◊

The following is a very important lemma that characterize the relations among min-entropy, conditional min-entropy and average min-entropy.

Lemma 7.1.6 (Relations among Entropies [DORS03, DRS04])**.** Let A, B, C be random variables. Then

(a) For any $\delta > 0$, the following holds with probability (over the choice of b) at least $(1-\delta)$:

$$\mathsf{H}_\infty(A \mid B = b) \geq \widetilde{\mathsf{H}}_\infty(A \mid B) - \log\left(\frac{1}{\delta}\right) \tag{7.7}$$

(b) If B has at most 2^λ possible values, then

$$\widetilde{\mathsf{H}}_\infty\big(A \mid (B, C)\big) \geq \widetilde{\mathsf{H}}_\infty\big((A, B) \mid C\big) - \lambda \geq \widetilde{\mathsf{H}}_\infty(A \mid C) - \lambda. \tag{7.8}$$

In particular,

$$\widetilde{\mathsf{H}}_\infty(A|B) \geq \mathsf{H}_\infty\big((A, B)\big) - \lambda \geq \mathsf{H}_\infty(A) - \lambda. \tag{7.9}$$

$\Diamond$

Lemma 7.1.7 (Generalized LHL [DORS03, DRS04])**.** Let $\mathcal{H} = \{h_i\}_{i\in\mathcal{I}}$ be a universal hash family[2] $\mathcal{I} \times \mathcal{D} \to \mathcal{R}$ with $|\mathcal{D}| = 2^n$ $|\mathcal{R}| = 2^\ell$ for some $n, \ell > 0$. Let $\mathsf{Ext}(x, i) = h_i(x)$. For any random variable X on support $\mathcal{D}$ and Y, the following holds:

$$\Delta\Big(\big(\mathsf{Ext}(X, U_\mathcal{I}), Y, U_\mathcal{I}\big), \big(U_\mathcal{R}, Y, U_\mathcal{I}\big)\Big) \leq 2^{-\left(\frac{\widetilde{\mathsf{H}}_\infty(X \mid Y) - \ell}{2} + 1\right)} \tag{7.10}$$

or equivalently (but easier to interpret),

$$\ell \leq \widetilde{\mathsf{H}}_\infty(X \mid Y) - c \quad \Rightarrow \quad \Delta\Big(\big(\mathsf{Ext}(X, U_\mathcal{I}), Y, U_\mathcal{I}\big), \big(U_\mathcal{R}, Y, U_\mathcal{I}\big)\Big) \leq 2^{-(\frac{c}{2}+1)} \tag{7.11}$$

where $U_\mathcal{I}$ and $U_\mathcal{R}$ are uniform distributions on $\mathcal{I}$ and $\mathcal{R}$ respectively.

In particular, to achieve statistical distance ε, we need to set

$$\ell \leq \widetilde{\mathsf{H}}_\infty(X \mid Y) - 2\log\big(\frac{1}{\varepsilon}\big) + 2$$

$\Diamond$

7.2 Randomness Extractors

Definition 7.2.1 (Randomness Extractor [NZ96])**.** Let the seed U_r be uniformly distributed on $\{0,1\}^r$. We say that a function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^\ell$ is a $(n, m, \ell, \varepsilon)$-strong extractor if, for all random variable X on $\{0,1\}^n$ with $\mathsf{H}_\infty(X) \geq m$, the following holds:

$$\Delta\Big(\big(\mathsf{Ext}(X, U_r), U_r\big), \big(U_\ell, U_r\big)\Big) \leq \varepsilon$$

$\Diamond$

Remark 7.2.2 (Strong vs. Standard Extractor)**.** Note that the extractor defined here is called *strong* extractor. The "standard" extractor only requires that $\mathsf{Ext}(X, U_r)$ is close to uniform. The

[2] For the definition of universal hash family, refer to Definition 6.3.1.

above version is called "strong" as it additionally requires the U_d part to be public. Usually, the strong version here is more widely used in cryptography.

Definition 7.2.3 (Average-Case Extractor [DORS03, DRS04])**.** Let the seed U_r be uniformly distributed on $\{0,1\}^r$. We say that a function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^\ell$ is an average-case $(n, m, \ell, \varepsilon)$-strong extractor if, for all pairs of random variables (X, Y) such that X has support $\{0,1\}^n$ and $\widetilde{\mathsf{H}}_\infty(X \mid Y) \geq m$, the following holds:

$$\Delta\Big(\big(\mathsf{Ext}(X, U_r), Y, U_r\big), \big(U_\ell, Y, U_r\big)\Big) \leq \varepsilon$$

$\Diamond$

Theorem 7.2.4 (Worst-Case to Average-Case Extractors [DORS03, DRS04])**.** For any $\delta > 0$, if Ext is a $(n, m - \log(\frac{1}{\delta}), \ell, \varepsilon)$-strong extractor, then Ext is also an average-case $(n, m, \ell, \varepsilon + \delta)$-strong extractor. $\Diamond$

Proof. The proof trivially follows from Lemma 7.1.6-(a). ■

Remark 7.2.5 (Interpreting LHL in term of Extractors)**.** By simple calculations on the parameters, one can interpret LHL in the following way:

- LHL says that universal hash families are $(n, m, \ell, \varepsilon)$-strong randomness extractors whenever $\ell \leq m - 2\log\left(\frac{1}{\varepsilon}\right) + 2$.
- Generalized LHL says that universal hash families are average-case $(n, m, \ell, \varepsilon)$-strong randomness extractors whenever $\ell \leq m - 2\log\left(\frac{1}{\varepsilon}\right) + 2$.

7.3 Expander Graphs

(to do..)

Chapter 8

Lattices

Many parts of this Chapter is taken from the marvelous survey of Peikert [Pei15]. I only pick the basic and widely-used materials. For an advanced and complete discussion on this topic, refer to [Pei15].

8.1 Basic Concepts

Dual Lattices. Given a lattice $\mathcal{L}$, it is easy to see that the set of points whose inner products with the vectors in $\mathcal{L}$ are all integers constitutes a lattice. Such a lattice is called dual lattice of $\mathcal{L}$, usually denoted as $\mathcal{L}^*$.

Definition 8.1.1 (Dual Lattice)**.** The dual (sometimes called reciprocal) of a lattice $\mathcal{L} \subseteq \mathbb{R}^n$ is defined as:

$$\mathcal{L}^* = \{\boldsymbol{v} : \langle \boldsymbol{v}, \mathcal{L} \rangle \subseteq \mathbb{Z}\}$$

Moreover, if $\boldsymbol{B}$ is a basis of $\mathcal{L}$, then $\boldsymbol{B}^{-\mathrm{T}} = (\boldsymbol{B}^{-1})^{\mathrm{T}} = (\boldsymbol{B}^{\mathrm{T}})^{-1}$ is a basis of $\mathcal{L}^*$. $\Diamond$

For example, $(c\mathcal{L})^* = c^{-1}\mathcal{L}$.

8.2 Computational Problems on Lattices

8.2.1 The Shortest Vector Problem

The most basic and important problem on lattices is the shortest Vector Problem (SVP). This problem has been here since 18th century, attracting attentions from famous mathematicians including Gauss and Minkovski.

Definition 8.2.1 (Shortest Vector Problem)**.** Given an arbitrary basis $\boldsymbol{B}$ of some lattice $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$, find a shortest nonzero lattice vector, i.e., a $\boldsymbol{v} \in \mathcal{L}$for which $\|\boldsymbol{v}\|_2 = \lambda_1(\mathcal{L})$. $\Diamond$

This question has been open for hundreds of years. But until today, we still do not have a solution. One important result for this question is the following theorem given by Minkovski, which upper-bounds the solution.

Theorem 8.2.2 (Minkowski's First Theorem)**.** For any lattice $\mathcal{L}$, we have $\lambda_1(\mathcal{L}) \leq \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$. $\Diamond$

There are several other theorems of this kind, e.g. Hermite's Theorem, Gauss Heuristic. See [HPSS08] for more interesting materials.

Although the SVP problem is very fascinating, more closely related to modern cryptography is the approximate version of SVP (and also some other problems on lattices of similar flavor). We now summarize them in the following.

Definition 8.2.3 (Approximate SVP Problem)**.** For lattice dimension parameter n, $\mathsf{gapSVP}_{\gamma(n)}$ is a promise (decisional) problem. On input $(\mathcal{L}, d)$, where $\mathcal{L}$ is a n-dimensional lattice and d is real number, output:

- YES: if $\lambda_1(\mathcal{L}) \leq d$,
- NO: if $\lambda_1(\mathcal{L}) > \gamma(n) \cdot d$

$\lozenge$

Definition 8.2.4 (Approximate Shortest Independent Vector Problem)**.** Given a basis $\boldsymbol{B}$ of a full-rank n-dimensional lattice $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$ (i.e. $\boldsymbol{B}$ is a $n \times n$ full-rank matrix), output a set $S = \{s_i\} \subset \mathcal{L}$ of n linearly independent lattice vectors where $\|s_i\|_2 \leq \gamma(n) \cdot \lambda_n(L)$ for all i. $\lozenge$

Definition 8.2.5 (Bounded-Distance Decoding)**.** Given a basis $\boldsymbol{B}$ of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$ and a target point $t \in \mathbb{R}^n$ with the guarantee that $\mathsf{dist}(t, \mathcal{L}) < d = \lambda_1(\mathcal{L})/(2\gamma(n))$, the bounded-distance decoding problem BDD_γ is to find the unique lattice vector $v \in \mathcal{L}$ such that $\|t - \boldsymbol{v}\|_2 < d$. $\lozenge$

Algorithms and complexity.[1] The above lattice problems have been intensively studied and appear to be intractable, except for very large approximation factors. Known polynomial-time algorithms like the one of Lenstra, Lenstra, and Lovász [LLL82] and its descendants (e.g., [Sch87] with [AKS01] as a subroutine) obtain only slightly sub-exponential approximation factors $\gamma = 2^{\Theta(n \log\log n/\log n)}$ for all the above problems. Known algorithms that obtain polynomial $\mathsf{poly}(n)$ or better approximation factors, such as [Kan83, AKS01, MV10, ADRS15], either require super-exponential $2^{\Theta(n \log n)}$ time, or exponential $2^{\Theta(n)}$ time *and* space. There are also time-approximation tradeoffs that interpolate between these two classes of results, to obtain γ approximation factors in $2^{\widetilde{\Theta}(n/\log\gamma)}$ time [Sch87]. Importantly, the above also represents the state of the art for quantum algorithms, though in some cases the hidden constant factors in the exponents are somewhat smaller (see, e.g. [**?**]). By contrast, the integer factorization and discrete logarithm problem (in essentially any group) can be solved in polynomial time using Shor's quantum algorithm [Sho99].

On the complexity side, many lattice problems are known to be NP-hard (sometimes under randomized reductions), even to approximate to within various sub-polynomial $n^{o(1)}$ approximation factors. E.g., for the hardness of SVP, see [Ajt98, Mic98, Kho04, HR07]. However, such hardness is not of any direct consequence to cryptography, since lattice-based cryptographic constructions so far rely on polynomial approximation problems factors $\gamma(n) \geq n$. Indeed, there is evidence that for factors $\gamma(n) \geq \sqrt{n}$, the lattice problems relevant to cryptography are not NP-hard, because they lie in $\mathsf{NP} \cap \mathsf{co}\mathcal{NP}$ [GG98, AR04].

[1]this part is taken verbatim from [Pei15]

8.2.2 Short Integer Solution (SIS)

Short Integer Solution. The short integer solution (SIS) problem was first introduced in the seminal work of Ajtai [Ajt96], and has served as the foundation for one-way and collision-resistant hash functions, identification schemes, digital signatures, and other "minicrypt" primitives (but not public-key encryption).

Definition 8.2.6 (Short Integer Solution)**.** For $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, the short integer solution problem $\mathsf{SIS}_{n,q,\beta,m}$ asks to find a nonzero integer vector $\boldsymbol{z} \in \mathbb{Z}^m$ of norm $\|\boldsymbol{z}\|_2 \leq \beta$ such that

$$\boldsymbol{A}\boldsymbol{z} = \boldsymbol{0} \in \mathbb{Z}_q^n$$

$\Diamond$

Theorem 8.2.7 (Hardness of SIS)**.** For any $m = \mathsf{poly}(n)$, any $\beta > 0$, and any sufficiently large $\beta \leq q/\mathsf{poly}(n)$, solving $\mathsf{SIS}_{n,q,\beta,m}$ with non-negligible probability is at least as hard as solving gapSVP_γ and SIVP_γ on arbitrary n-dimensional lattices (i.e., in the worst case) with overwhelming probability, for some $\gamma = \beta \cdot \mathsf{poly}(n)$. $\Diamond$

Here are some remarks on the harness parameters:

- For $\beta \geq q$, the SIS problem is easy: simply setting $\boldsymbol{z} = (q, 0, \dots, 0)^{\mathrm{T}}$ gives us $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{0} \bmod q$.
- β and m have to be large enough to guarantee the existence of a solution. This is the case whenever $\beta \geq \sqrt{n \log q}$ and $m \geq n \log q$. This is because of the following pigeonhole argument: first, we can assume without loss of generality that $m = n \log q$.[2] Then because there are more than q^n vectors $x \in \{0,1\}^m$, there must be two distinct x, x' such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x'} \in \mathbb{Z}_q^n$, so their difference $z = x - x' \in \{0, 1, -1\}^m$ is a solution with $\|\boldsymbol{z}\|_2 \leq \beta$ for $m = n \log q$ and $\beta \geq \sqrt{n \log q}$.
- After a line of work[MR04, GPV08], the state-of-the-art value for the hardness parameters are $\gamma = \beta \cdot \widetilde{O}(\sqrt{n})$ and $\beta \leq q/\widetilde{O}(\sqrt{n})$.
- [MP13] achieve $\beta \leq q/n^\varepsilon$ for any constant $\varepsilon > 0$. But the γ is somewhat subtle: it can depend on the norm of the SIS solution in the ℓ_∞ norm.

8.2.3 Ring-SIS

In the standard SIS problem defined in previous section, the underlying sets sets are $\mathbb{Z}^n$ and $\mathbb{Z}_q^n$. Roughly speaking, Ring-SIS is the ring version of the standard SIS problem, i.e. the underlying sets are rings R and R_q (corresponding to $\mathbb{Z}^n$ and $\mathbb{Z}_q^n$ in the standard SIS setting, respectively). People care bout this ring version because it usually provides more efficient cryptographic constructions, due to the "richer" algebraic structure of the underlying rings. Of course, the analysis of harness assumptions requires more careful analysis, as the "richer" structures of rings admits more attacks than that for a standard SIS.

[2] This is because once we can solve SIS for $\boldsymbol{A}_{n\times m}$, we can easily extend the solution when more rows are appended at the end of $\boldsymbol{A}$: simply append 0's at the end of the solution vector.

Definition 8.2.8 (Ring-SIS)**.** Let R be a ring, equipped with some norm $\|\cdot\|$. For a positive integer q, denote the quotient ring R/qR as R_q. For $\boldsymbol{a} \overset{\$}{\leftarrow} R_q^m$, the Ring-SIS problem $\mathsf{R\text{-}SIS}_{q,m,\beta}$ is to find a nonzero vector $\boldsymbol{z} \in R^m$ of norm $\|\boldsymbol{a}\| \leq \beta$ such that:

$$F_{\boldsymbol{a}}(\boldsymbol{z}) = \langle \boldsymbol{a}, \boldsymbol{z} \rangle = 0 \in R_q.$$

$\diamondsuit$

Remark 8.2.9 (Harness of Ring-SIS)**.** The hardness of Ring-SIS depends on the choice of the underlying ring R and the norm $\|\cdot\|$. A typical choice is to set R to be the so-called *rank-n ring of convolution polynomials* $\mathbb{Z}[x]/\langle x^n - 1\rangle$, in which case R_q will be $\mathbb{Z}_q[x]/\langle x^n - 1\rangle$.

When the ring is of the form $\mathbb{Z}[x]/\langle f(x)\rangle$ where $\mathsf{deg}(f) = n$, the preferred norm is the so-called *canonical embedding* $\sigma : \mathbb{Z}[x]/\langle f(x)\rangle \to \mathbb{C}^n$ from algebraic number theory. This embedding maps each ring element $r \in R$ to the vector $(r(\alpha_1), \ldots, r(\alpha_n)) \in \mathbb{C}^n$, where the $\alpha_i \in \mathbb{C}$ are the n complex roots of $f(X)$.

Such choice of the underlying ring and the norm has several advantages. As the reason is advanced and complicate, we do not provide further discussion. We refer the readers to Section 4.3 in [Pei15].

8.2.4 Learning with Error (LWE)

The learning with errors (LWE) problem was defined by Regev [Reg05].

Definition 8.2.10 (Decisional LWE Problem [Reg05])**.** The $\mathsf{LWE}_{n,q,\chi,m}$ problem is two distinguish the following tow distributions:

$$(\boldsymbol{A}, \boldsymbol{s}^{\mathrm{T}}\boldsymbol{A} + \boldsymbol{e}^{\mathrm{T}} \bmod q) \text{ and } (\boldsymbol{A}, \boldsymbol{u}^{\mathrm{T}})$$

where $\boldsymbol{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n\times 1}$, $\boldsymbol{e} \leftarrow \chi^{n\times 1}$ and $\boldsymbol{u} \overset{\$}{\leftarrow} \mathbb{Z}_q^m$. $\diamondsuit$

Different presentations of the hardness reduction of (average-case) LWE assumption to (worst-case) lattice problems exist in the literature. The one presented here (taken from [GSW13]) is probably the clearest one.

Definition 8.2.11 (B-Bounded Dsitributions)**.** A distribution ensemble $\{\chi_n\}_{n\in\mathbb{N}}$, supported over the integers, is called B-bounded if

$$\Pr_{e\leftarrow\chi_n}[|e| > B] \leq \mathsf{negl}(n)$$

$\diamondsuit$

The following theorem shows the reduction from the LWE problem to the GapSVP problem, which is critical for all LWE-based cryptosystem. This idea is originated from [Reg05], and refined in [Pei09, MM11, MP12]. The version presented here is stated as Corollary 2.1 from [Bra12].

Theorem 8.2.12 (Hardness of LWE)**.** Let $q = q(n) \in \mathbb{N}$ be either a prime power or a product of small (size $\mathsf{poly}(n)$) distinct primes, and let $B \geq \omega(\log n) \cdot n$. Then there exists an efficient sampleable B-bounded distribution χ such that if there is an efficient algorithm that solves the average-case LWE problem for parameters n, q, χ, then:

- There is an efficient *quantum* algorithm that solves $\mathsf{gapSVP}_{\widetilde{O}(nq/B)}$ on any n-dimensional lattice.

- If $q \geq \widetilde{O}(2^{n/2})$, then there is an efficient *classical* algorithm for $\mathsf{gapSVP}_{\widetilde{O}(nq/B)}$ on any n-dimensional lattice.

In both cases, if one also considers distinguishers with sub-polynomial advantage, then we require $B \geq \widetilde{O}(n)$ and the resulting approximation factor is slightly larger than $\widetilde{O}(n^{1.5}q/B)$. $\Diamond$

Modulus-to-Noise Ratio. The value q/B usually arouses concerns regarding the efficiency of constructions , so people refer to it as "modulus-to-noise ratio".

Discrete Gaussian Distribution. The most widely-used error distribution to construct hard LWE problem is the discrete version of Gaussian distribution. It is the distribution over $\mathbb{Z}$ where the probability of x is proportional[3] to $e^{-\pi(|x|/\sigma)^2}$, where σ is the width parameter. The hardness of LWE w.r.t. discrete Gaussian distribution is stated as the following theorem. A discrete Gaussian with parameter σ is $B = \sigma$ bounded, except with negligible probability.

Theorem 8.2.13 (Hardness of LWE w.r.t. Discrete Gaussian [Reg05])**.** For any $m = \mathsf{poly}(n)$, any modulus $q \leq 2^{\mathsf{poly}(n)}$, and any (discretized) Gaussian error distribution χ of parameter $\sigma = \alpha \cdot q \geq 2\sqrt{n}$ where $0 < \alpha < 1$, solving the decisional $\mathsf{LWE}_{n,q,\chi,m}$ problem is at least as hard as quantumly solving $\mathsf{gapSVP}_{\widetilde{O}(n/\alpha)}$ and $\mathsf{SIVP}_{\widetilde{O}(n/\alpha)}$ on arbitrary n-dimensional lattices. $\Diamond$

Note that the exact values of m (the number of samples) and q (the modulus) play essentially no role in the ultimate hardness guarantee (apart from the lower bound for $q \geq 2\sqrt{n}/\alpha$). However, the approximation factor $\gamma = \widetilde{O}(n/\alpha)$ degrades with the modulus-to-noise ration $\sigma/q = 1/\alpha$. For gapSVP_γ and SIVP_γ, the best known (classical or quantum) algorithms for these problems run in time $2^{\widetilde{O}(n/\log\gamma)}$, and in particular they are conjectured to be intractable for $\gamma = \mathsf{poly}(n)$.

8.2.5 Learning with Rounding

LWE problem is inherently randomized. But there are some crypto primitives (e.g. PRF) that prefers deterministic hardness assumption. To address this issue, [BPR12] proposed a deterministic version of LWE, called "learning with rounding" (LWR), and showed how to reduce it to LWE. LWR is used to build efficient PRF based on hard lattice problems, and plays a important role in other applications such as watermarking [KW17, KW19] and trapdoor hash functions [DGI+19].

Definition 8.2.14 (Rounding Function)**.** For integers $p \geq q \geq 2$, the rounding function $\lfloor \cdot \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ is defined as:

$$\lfloor x \rceil_p = \left\lfloor \frac{(x \ \mathsf{mod}\ q)}{q} \cdot p \right\rceil \ \mathsf{mod}\ p$$

This notion extends to vectors and matrices component-wisely. $\Diamond$

[3] "Proportional" means that one needs to normalize the value such that the probability for each $x \in \mathbb{Z}$ sum up to 1.

Definition 8.2.15 (Learning with Rounding)**.** For a distribution D_s on $\mathbb{Z}_q^{n\times 1}$, the learning with rounding problem $\mathsf{LWR}_{n,q,p,m}^{D_s}$ problem is two distinguish between the following tow distributions:

$$(\boldsymbol{A}, \lfloor \boldsymbol{s}^{\mathrm{T}}\boldsymbol{A}\rceil_p) \text{ and } (\boldsymbol{A}, \boldsymbol{u}^{\mathrm{T}})$$

where $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{s} \xleftarrow{D_s} \mathbb{Z}_q^{n\times 1}$ and $u \xleftarrow{\$} \mathbb{Z}_p^{m\times 1}$ ◊

Definition 8.2.16 (Hardness of LWR)**.** Let χ be any efficiently sampleable B-bounded distribution over $\mathbb{Z}$, and let $p \leq \frac{q}{B\cdot n^{\omega(1)}}$. Then for any distribution D_s on $\mathbb{Z}_q^n$, solving decision $\mathsf{LWR}_{n,q,p,m}^{D_s}$ is at least as hard as solving decision $\mathsf{LWE}_{n,q,\chi,m}^{D_s}$, i.e. the $\mathsf{LWE}_{n,q,\chi,m}$ problem where the secret vector $\boldsymbol{s}$ comes from the same distribution D_s. ◊

8.2.6 Ring-LWE

Just like SIS vs Ring-SIS, the LWE problem also has a ring version called Ring-LWE.

Definition 8.2.17 (Decisional Ring-LWE Problems)**.** The decisional Ring-LWE problem $\mathsf{R\text{-}LWE}_{q,\chi,m}$ is to distinguish the following two distributions:

$$(\boldsymbol{a}, s\cdot \boldsymbol{a} + \boldsymbol{e} \bmod q) \text{ and } (\boldsymbol{a}, \boldsymbol{u})$$

where $\boldsymbol{a} \xleftarrow{\$} R_q^m$, $s \xleftarrow{\$} R_q$, $\boldsymbol{u} \xleftarrow{\$} R_q^m$ and $\boldsymbol{e} \leftarrow \chi^m$. ◊

As that case of Ring-SIS, the hardness of Ring-LWE depends on the choice of the underlying ring and the error distribution. This was investigated in the work of [LPR10], where they pick a *cyclotomic* ring and a special error distribution[4]. We will not provide further discussion here. Next, we will only list the hardness reduction theorem from Ring-LWE to gapSVP. We refer the readers to [LPR10, LPR13, AP13] and Section 4.4 of [Pei15] for more details.

Theorem 8.2.18 (Hardness of Ring-LWE [LPR10])**.** For any $m = \mathsf{poly}(n)$, cyclotomic ring R of degree n (over $\mathbb{Z}$), and appropriate choices of modulus q and error distribution χ of error rate $\alpha < 1$, solving the $\mathsf{R\text{-}LWE}_{q,\chi,m}$ problem is at least as hard as quantumly solving the gapSVP_γ problem on arbitrary ideal lattices in R, for some $\gamma = \mathsf{poly}(n)/\alpha$. ◊

8.3 Two Critical Equations for Lattice-Based Crypto

The materials presented in this part is based on the excellent talks by Hoeteck Wee (link) and David Wu (link).

To do...

Explain how to derive and apply the follow two equations

$$\mathbf{C}_1, \ldots, \mathbf{C}_n \mapsto \mathbf{C}_{f(x)}$$
$$[\mathbf{C}_1 - x_1\mathbf{G} \mid \ \ldots \ \mid \mathbf{C}_n - x_n\mathbf{G}]\mathbf{H}_{f,x} = \mathbf{C}_f - f(x)\mathbf{G}$$

[4]Actually, [LPR10] uses a certain fractional ideal $R^\vee$ that is dual to R

Chapter 9

Coding Theory

9.1 Basic Concepts

Definition 9.1.1. An $[n,k,d]_q$ code is a function $C : \Sigma^k \to \Sigma^n$ such that:

- $|\Sigma| = q$;
- For every $x, x' \in \Sigma^k$, $\mathsf{dist}_H(C(x), C(x')) \geq d$, where $\mathsf{dist}_H(\cdot,\cdot)$ is the hamming distance.

$\Diamond$

(talk about code rate (or information rate) R. Fractional Hamming distance. δ-distance code.)

9.2 The Bounds

Lemma 9.2.1 (Singleton Bound). Let C be a $[n,k,d]_q$ code. Then $k \leq n - d + 1$ $\Diamond$

Proof. Let $C' : \Sigma^k \to \Sigma^{n-d+1}$ be the projection of C to the first $n-d+1$ coordinates. That is, $C'(x)$ contains the first $n-d+1$ entries of $C(x)$. We see that C' must be an injective function, because if $C'(x) = C'(x')$ with $x \neq x'$, then $C(x)$ and $C'(x)$ can differ in at most $d-1$ coordinates, contradicting the fact that C has minimum distance at least d. But if C' is injective then its range must be at least as large as its domain, and so $n - d + 1 \geq k$. ■

Lemma 9.2.2 (Gilbert Bound). For every n and $\frac{d}{n} < \frac{1}{2}$ there is a $[n,k,d]_2$ code such that

$$k \geq n \cdot \left(1 - H_2(\frac{d}{n})\right) - \Theta(\log n)$$

where $H_2(\cdot)$ is Shannon's binary entropy.

In terms of code rate and δ distance, the Gilbert bound shows that for any $\delta < \frac{1}{2}$, when n is large enough, there always exists a $[n,k,d]_2$ code such that:

$$\mathsf{R} \geq 1 - H_2(\delta) - o(1)$$

$\Diamond$

Proof. To do the proof, we first need to recall some implications from Stirling's formula. Stirling's approximation gives

$$n! = \Theta\big(\sqrt{n} \cdot (\frac{n}{e})^n\big)$$

This implies:

$$\binom{n}{k} = \Theta\Bigg(\sqrt{\frac{n}{k(n-k)}} \cdot (\frac{n}{k})^k \cdot (\frac{n}{n-k})^{n-k}\Bigg)$$

which implies:

$$\log \binom{n}{k} = n \cdot H_2(\frac{k}{n}) + \Theta(\log n)$$

(to be done from Luca's Lecture notes...

Also, check the Gilbert-Varshamov bound in Chapter 19.2 in [AB09].) ■

9.3 Linear Code

Given a specific tuple of (n, k, d), there are so many ways to design a $[n, k, d]$ code. One special type of codes draw our attention due to its clean format and rich theoretical implications. Such kind of codes is linear code (on $\mathbb{F}_2$). Its linear characterization admits the application of the beautiful theory of linear algebra, while does not give up the power of error correction too much.

The codeword of a $[n, k, d]_2$ linear code can be treated as a dimension-k linear subspace of $\mathbb{F}_2^n$. Then any set basis $\{g_1, \ldots, g_k\}$ for the codeword space are called generators of this linear code. Any codewords in this space can be expressed as a matrix-vector multiplication $\mathbf{Cx}$, where $\mathbf{C}$ is a $k \times n$ whose i-th column is g_i. The *parity check matrix* is the matrix $\mathbf{H}$ such that $\mathsf{Ker}(\mathbf{H}) = \mathbf{C}$. It is also the generator matrix of $\mathbf{C}^\perp$, the dual code of $\mathbf{C}$. It has the property that a codeword $\mathbf{c}$ is in $\mathbf{C}$ if and only if $\mathbf{Hc} = 0$. Due to this property, the value $\mathbf{Hc}$ is called the "syndrome" of $\mathbf{c}$.

9.4 Walsh-Hadamard Code

Walsh-Hadamard code is a $[2^n, n, 2^{n-1}]_2$ code. Given any message $m \in \{0,1\}^n$

$$\mathsf{WH}(m) = (\langle m, [1]_2\rangle, \langle m, [2]_2\rangle, \ldots, \langle m, [2^n]_2\rangle),$$

where $[i]_2$ is the binary representation of i, and "$\langle \cdot, \cdot \rangle$" is the inner product modulo 2.

9.5 Reed-Solomon Code

Reed-Solomon code makes use of larger alphabet size to achieve better distance and rate. It is a $[n, k, n-k+1]_q$ code where:

- the alphabet is a size-q field $\mathbb{F}_q$; **and**
- $k \le n \le q$.

For a message $m = (m_0, \ldots, m_{k-1}) \in \mathbb{F}_q^k$, its codeword is computed as follows:

1. Treat $m = (m_0, \ldots, m_{k-1})$ as degree-$(k-1)$ polynomial in $\mathbb{F}_q[x]$:
$$m(x) = m_0 + m_1 \cdot x + m_2 \cdot x^2 + \ldots + m_{k-1} \cdot x^{k-1}.$$
2. Evaluate $m(x)$ on n prefixed points $\{x_1, \ldots, x_n\}$.
3. Output the evaluations as the codeword for m, i.e.
$$\mathsf{RS}(m) = \big(m(x_1), \ldots, m(x_n)\big).$$

Common choices for the set of evaluation points include $\{0, 1, \ldots, n-1\}$, $\{0, 1, \alpha, \alpha^2, \ldots, \alpha^{n-2}\}$, or $\{\alpha^0, \alpha^1, \ldots, \alpha^{n-1}\}$, where α is the primitive element of $\mathbb{F}_q$.

Vandermonde Matrix Representation. The encoding procedure of Reed-Solomon code can be expressed as a Vandermonde linear transformation. For example, if we use $\{x_1, \ldots, x_n\}$ as the set of evaluation points, then for any $m = (m_0, \ldots, m_{k-1})$,

$$\mathsf{RS}(m) = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{k-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{k-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{k-1} \end{bmatrix}_{n \times k} \cdot \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{k-1} \end{bmatrix}_{k \times 1} = \begin{bmatrix} m(x_1) \\ m(x_2) \\ \vdots \\ m(x_n) \end{bmatrix}_{n \times 1}$$

9.6 Coding theory in general

9.7 Non-malleable code

9.7.1 Splite-state non-malleable code

9.8 Randomized encoding (used in [KOS18])

Chapter 10

Complexity Theory

10.1 The Basics

Traditionally, Complexity Theory cares about constructible functions. Take time-constructible functions as an example (similar reason applies to space-constructible). For such functions, a TM "knows" the time bound under which it is operating, simply by "looking at" the description of the function. These functions are usually considered natural. Most importantly, several theorems only holds (provably) for such functions. A typical example is the time hierarchy theorem, whose proof requires that the TM must determine in $O(f(n))$ time whether an algorithm has taken more than $f(n)$ steps. Time-constructibility is thus proposed to formulate these natural functions.

Definition 10.1.1 (Time Constructibility)**.** A function $f : \mathbb{N} \to \mathbb{N}$ is time-constructible if there is a TM M that computes the function $1^n \to [f(n)]_2$ in $O\big(f(n)\big)$ time, where $[f(n)]_2$ denotes the binary representation of the number $f(n)$. $\Diamond$

Remark 10.1.2. Here are some remarks:

1. Usually, a Turing machine uses a binary alphabet. If we use such a TM to compute the function $1^n \to f(n)$, the output is by default in binary representation. In the above definition, we put $[f(n)]_2$ mainly to make this requirement explicit for the machines which do not use a binary alphabet. But this does not matter much since other alphabet can be converted into a binary one without much blowing-up in time complexity.

2. Some textbook define time constructibility only for functions $f(n) > n$. That is to allow the algorithm time to read its input.

3. There is a definition called "fully time-constructible functions". It is the same as Definition 10.1.1 except that the computation should be done in exactly $f(n)$ time, instead of $O\big(f(n)\big)$.

Definition 10.1.3 (Space Constructibility)**.** A function $f : \mathbb{N} \to \mathbb{N}$ is space-constructible if there is a TM that computes the function $1^n \to [f(n)]_2$ in $O(f(n))$ space, where $[f(n)]_2$ denotes the binary representation of the number $f(n)$. $\Diamond$

In some scenarios such as computing on low storage machines, the space resource can be a bottleneck of computation power. Thus people also care about the class of language captured by deterministic/non-deterministic logarithm space. Three potential problems arise when we want to investigate L and NL:

1. The input already occupies linear space.

2. The machine may not have enough space to write down the full output.

3. Since $\mathsf{NL} \subseteq \mathsf{P}$, NL may not be "closed" under Karp reduction.

For the first problem, we do not count the space occupied by the input. In addition, we usually (e.g. in Savitch's theorem 10.5.1) restrict ourselves to space complexity $f(n) \geq \log(n)$, such that we have enough space to write down the index of the input position that we want to access. For the last two problems, people propose implicitly-computable functions and log space reduction as shown in the following definitions.

Definition 10.1.4 (Implicit Logspace Computability)**.** A function $f : \{0,1\}^* \to \{0,1\}^*$ is implicitly logspace computable, if the following holds

(1) $\forall x \in \{0,1\}^*$, $\exists c$ s.t. $|f(x)| \leq |x|^c$ (i.e. f is polynomially bounded).

(2) The languages $L_f = \{(x,i) \mid f(x)_i = 1\}$ and $L'_f = \{(x,i) \mid i \leq |f(x)|\}$ are in L.

$\Diamond$

Remark 10.1.5. The definition of logspace computability may seem confusing at first glance.

1. The definition of logspace computability may seem confusing at first glance. But all it wants to say is that the function can be computed using log space. This requirement boils down to the two languages in the second item of the definition: (i) $L_f \in \mathsf{L}$ means each bit of the output can be computed in log space; (ii) $L'_f \in \mathsf{L}$ means the total length of the output can be computed in log space. Also, note that the first item in the definition is to restrict us to functions the polynomial output size. Without it, a function with exponential size of output may also satisfy the requirement in (2).

2. Do not confuse it with the concept of space/time constructible functions (Definition 10.1.1 and 10.1.3). Indeed, space/time constructibility is more about the properties of functions, instead of machines. It is proposed to capture all the "interesting" and natural functions people care about, ruling out functions that are troublesome to analysis (Fortunately, those cases are usually rare and unnatural). In contrast, implicit-logspace computability is more about the machines. Since logspace machine do not have enough space to write down the full output, people thus propose this class of function to allow meaningful discussion for such machines.

Definition 10.1.6 (Logspace Reduction and NL-Completeness)**.** A language A is logspace reducible to language B, denoted $A \leq_{\mathsf{log}} B$, if there exists a implicitly logspace computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for all $x \in \{0,1\}^*$,

$$x \in A \quad \Leftrightarrow \quad f(x) \in B$$

We say that $B \in \mathsf{NL}$ is NL-complete if for every $A \in \mathsf{NL}$, $A \leq_{\mathsf{log}} B$. $\Diamond$

Definition 10.1.1: Various Types of Reductions

Karp Reduction: let X and Y be decisional problems. A polynomial-time computable function f is called a Karp reduction from X to Y if, for every x, it holds that $x \in X$ if and only if $f(x)\ inY$.

The following quote from [Gol08] explains the relation between Karp reduction and Turing (aka Cook) reduction: "Thus, syntactically speaking, a Karp-reduction is not a Cook-reduction, but it trivially gives rise to one (i.e., on input x, the oracle machine makes query $f(x)$, and returns the oracle answer). Being slightly inaccurate but essentially correct, we shall say that Karp-reductions are special cases of Cook-reductions."

Levin Reduction: let R and R' be relations for two search problems. Let

$$S_R = \{x : \exists y \text{ s.t. } (x,y) \in R\}, \quad S_{R'} = \{x' : \exists y' \text{ s.t. } (x',y') \in R'\}.$$

A pair of polynomial-time computable functions, f and g, is called a Levin reduction from R to R' if f is a Karp reduction from S_R to $S_{R'}$, and for every $x \in S_R$ and $y' \in R'(f(x))$ it holds that $\big(x, g(x,y')\big) \in R$, where $R'(x') = \{y' : (x',y') \in R'\}$.

Levin Reduction can be viewed as a generalization of Karp reduction to search problems. We will use this type of reduction in Theorems 10.4.1 and 10.6.2.

Turing Reductions: if A is Turing reducible to B in polynomial time, then $A \subseteq P^B$. This reduction is usually denoted as $A \leq_T^p B$. This type of reduction is also called "**Cook reduction**". It is useful for $\#P$ completeness.

Parsimonious Reductions: a Parsimonious reduction R from problem X to problem Y is a reduction such that for any instance x of X, the number of solutions to x is equal to the number of solutions to problem $R(x)$, which is an instance of Y.

Theorem 10.1.7 (Efficient Universal Turing Machine [HS66])**.** There exists a TM U such that for every $x, \alpha \in \{0,1\}^*$, $U(x,\alpha) = M_\alpha(x)$, where M_α denotes the TM represented by α. Moreover, if M_α halts on input x within T steps then $U(x,\alpha)$ halts within $c \cdot T \log T$ steps, where $c > 0$ is a number depending only on

- M_α's alphabet size
- M_α's number of tapes
- M_α's number of states

$\Diamond$

10.1.1 Oblivious TM, Configuration Graphs and Snapshots

(To be done ...)

10.1.2 Transformations between Different Computational Models

Need to formalize these folklore claims

- Any Boolean circuit can be transformed into an equivalent arithmetic circuit over any field, with at most a constant-factor blowup in size.
- Any circuit Cir is not layered it can easily be transformed into a layered circuit Cir$'$ with a small blowup in size.
- If a computer program runs in time $T(n)$ on a RAM with at most $S(n)$ cells of memory, then the program can be turned into a (layered, fan-in 2) arithmetic circuit of depth not much more than $T(n)$ and width of about $S(n)$. [Each layer of the circuit represent a configuration of the RAM execution. So the circuit has depth $\approx T$ and width $\approx S$.]
- There exist a transformation from the time-T space-S RAM computation to a circuit of depth $S \log T$ and size $2^{\Theta(S)}$. (See [Tha20, Section 5.4].)
- **RAM-to-CirSAT.** Any RAM program, running in time T and outputting y on input x, can be efficiently transformed into an instance (C, x, y) of arithmetic circuit satisfiability (i.e. $\exists w$ s.t. $C(x, w) = y$), where the circuit C has size close to T and depth close to $\log T$, and the witness w is of size $\approx T$. (See [Tha20, Section 5.5].)

10.2 Boolean Circuits

Definition 10.2.1: Boolean Circuits

For every $n \in \mathbb{Z}$, an n-input single-output Boolean circuit is a directed acyclic graph with n sources (vertices with no incoming edges) and one sink (vertex with no outgoing edges). All non-source vertices are called gates and are labeled with one of $\wedge$, $\vee$ or $\neg$ (i.e., the logical operations OR, AND, and NOT). The vertices labeled with $\vee$ and $\wedge$ have fan-in (i.e., number of incoming edges) equal to 2, and the vertices labeled with $\neg$ have fan-in 1. The size of Cir, denoted by $|\mathsf{Cir}|$, is the number of vertices in it.

Remark 10.2.2: On the number of fan-out

Note that, in Definition 10.2.1, we do not put any restriction on the fan-out of sources, i.e. one input can go to several gates. But, traditionally, other gates (except for the input gates) have fan-out 1.
It does not make too much difference to allow more fan-out. However, the fan-in is usually clearly stipulated as in Definition 10.2.1. Two remarks follow:

- For circuits whose fan-in ≤ 2, its number of edges cannot be too big even if arbitrary fan-out number is allows. Assume that the number of total gates is m for such circuit. Then the total number of its edges is bounded by $2m$.
- Boolean formulae are just circuits where each gate has fan-in equal to 1.

Fact 10.2.1. Here are some simple but useful facts w.r.t. Boolean circuits:

- NAND-gate circuits are universal. A NAND-gate is defined as: $\mathsf{NAND}(a, b) = \neg(a \wedge b) = \neg a \vee \neg b$. We can convert a AND and OR gate to a NAND gate in the following way:

    1. $\mathsf{NOT}(a) = \mathsf{NAND}(a, a)$
    2. $\mathsf{AND}(a, b) = \neg\mathsf{NAND}(a, b) = \mathsf{NAND}\big(\mathsf{NAND}(a, b), \mathsf{NAND}(a, b)\big)$
    3. $\mathsf{OR}(a, b) = \mathsf{NAND}(\neg a, \neg b) = \mathsf{NAND}\big(\mathsf{NAND}(a, a), \mathsf{NAND}(b, b)\big)$

- $\mathsf{NAND}(a, b) = c$ if and only if $a + b + 2c - 2 \in \{0, 1\}$. This simple obervation helps in building NIZK and NIWI in [GOS06b, GOS06a].

Theorem 10.2.2 (Boolean Formula to CNF)**.** For every Boolean function $f : \{0, 1\}^\ell \to \{0, 1\}$ there is an ℓ-variable CNF formula ϕ of size $\ell \cdot 2^\ell$ such that $\phi(x) = f(x)$ for every $x \in \{0, 1\}^\ell$, where the size of a CNF formula is defined to be the number of $\wedge$ or $\vee$ symbols it contains. $\diamondsuit$

Proof. We give a constructive proof, building such a CNF formula ϕ explicitly. For a variable $v \in \{0, 1\}^\ell$, it is easy to construct a ℓ-variable "characteristic function" $C_v(z_1, \ldots, z_\ell)$ such that $C_v(z_1, \ldots, z_\ell)$ only consists of disjunctions among z_i's and $\overline{z}_i$'s, and satisfies the following requirement:

$$C_v(z_1, \ldots, z_\ell) = \begin{cases} 1, & \text{if } z_1 \| \ldots \| z_\ell = v \\ 0, & \text{if } z_1 \| \ldots \| z_\ell \neq v \end{cases}$$

Then the following formula satisfies all the property specified in the theorem:

$$\phi(z_1, \ldots, z_\ell) = \bigwedge_{v : f(v) = 1} C_v(z_1, \ldots, z_\ell)$$

■

The following theorem is not surprising, given that 3SAT is NP complete. Also, the proof is not hard. However, its proof contains very useful tricks for converting circuit gates to formula clauses (more accurately, 3CNF clauses).

Theorem 10.2.3: CKT-SAT to 3CNF

There is a polynomial-time reduction from CKT-SAT to 3CNF. (This theorem is w.r.t. 2-fan-in, unbounded fan-out circuits.)

Proof. (See the second half of this video.

The key trick is the following formula from mathematical logics:

$$a \Rightarrow b \quad \Leftrightarrow \quad \neg a \vee b.$$

Using this formula, we get that

- For AND gates: $w_3 = w_1 \wedge w_2$ can be converted to $(\neg w_3 \vee w_1) \wedge (\neg w_3 \vee w_2) \wedge (\neg w_1 \vee \neg w_2 \vee w_3)$. The proof goes as follows:

$$\begin{aligned} & w_3 = w_1 \wedge w_2 \\ \Leftrightarrow\ & (w_3 \Leftrightarrow w_1 \wedge w_2) \\ \Leftrightarrow\ & (w_3 \Rightarrow w_1 \wedge w_2) \wedge (w_1 \wedge w_2 \Rightarrow w_3) \\ \Leftrightarrow\ & \big(\neg w_3 \vee (w_1 \wedge w_2)\big) \wedge \big(\neg(w_1 \wedge w_2) \vee w_3\big) \\ \Leftrightarrow\ & (\neg w_3 \vee w_1) \wedge (\neg w_3 \vee w_3) \wedge (\neg w_1 \vee \neg w_2 \vee w_3) \end{aligned}$$

- For NOT gates: $w_2 = \neg w_1$ can be converted to $(w_1 \vee w_2) \wedge (\neg w_1 \vee \neg w_2)$. Its proof goes as the above.
- For OR gates: $w_3 = w_1 \vee w_2$ can be converted to $(\neg w_3 \vee w_1 \vee w_2) \wedge (\neg w_1 \vee w_3) \wedge (\neg w_2 \vee w_3)$. Its proof goes as the above.

More generally, any propositional formula involving 3 variables can be converted to a CNF.) ■

10.2.1 Interesting Classes of Circuit Complexity

Definition 10.2.3 (The class NC)**.** For every d, a language $\mathcal{L}$ is in NC^d if $\mathcal{L}$ can be decided by a family of Boolean circuits $\{\mathsf{Cir}_n\}$ where Cir_n satisfies the following requirements:

- it is of $\mathsf{poly}(n)$ size, **and**
- it is of $O(\log^d n)$ depth.

The class NC is $\cup_{i\geq 0}\mathsf{NC}^i$. ◊

Definition 10.2.4 (The class AC)**.** For every d, a language $\mathcal{L}$ is in AC^d if $\mathcal{L}$ can be decided by a family of Boolean circuits $\{\mathsf{Cir}_n\}$ where Cir_n satisfies the following requirements:

- it is of $\mathsf{poly}(n)$ size, **and**
- it is of $O(\log^d n)$ depth, **and**
- its OR and AND gates are allowed to have unbounded fan-in.

The class AC is $\cup_{i\geq 0}\mathsf{AC}^i$. ◊

Fact 10.2.5. $\mathsf{NC}^i \subseteq \mathsf{AC}^i \subseteq \mathsf{NC}^{i+1}$, where the inclusion is known to be strict for $i = 0$. (Proof: Unbounded (but $\mathsf{poly}(n)$) fan-in can be simulated using a tree of OR/AND gates of depth $O(\log n)$.)

Fact 10.2.6. The following problems (more accurately, their language version) are in AC^0:

1. Binary addition (with carry bits) of 2 n-bit binary strings.

Here are some interesting lower-bounds for AC^0:[1]

1. Depth-2 circuits are either DNFs or CNFs.

[1]for more details, check this lecture note

2. Every Boolean function can be computed by a (exponential-size) DNF and also by a CNF. (This is just Thm. 10.2.2.)

3. Depth-2 AC^0 circuits for PARITY have size at least $\Omega(2n)$.

4. If Cir is an AC^0 circuit of size s, depth d computing PARITY, then $s \geq 2^{\Omega(n^{\frac{1}{d-1}})}$.

   We remark that this famous theorem says AC^0 circuit of constant depth and subexponential size[2] cannot compute PARITY. It is a highly non-trivial result. The proof exploits Håstad's switching lemma.

The following problems (more accurately, their language version) are in NC^0:

1. Binary addition (with carry bits) of 3 n-bit binary strings.

The following problems (more accurately, their language version) are in NC^1:

1. Binary addition (with carry bits) of n length n-bit binary strings.

2. Integer multiplication (of 2 n-bit numbers).

3. Integer division.

4. Inner product.

5. Matrix Multiplication (of 2 $n \times n$ matrices where each entry is of size n).

6. Parity checking: $\mathsf{PARITY} = \{x : x \text{ has an odd number of 1s}\}$

7. Evaluation of polynomial size Boolean formulae.

The following problems (more accurately, their language version) are in NC^2:

1. $\boldsymbol{A}^n$ of $n \times n$ matrix $\boldsymbol{A}$ where each entry of size n.

2. Determinant of $n \times n$ matrix.

3. Solving the linear system $\boldsymbol{Ax} = \boldsymbol{b}$, where A is a non-singular $n \times n$ matrix, b an n dimensional column vector. (The algorithm is non-trivial. See this lecture notes.)

10.2.2 Branching Program

Theorem 10.2.7 (Barrington's Theorem)**.** (To do ...) $\Diamond$

[2]Note that some authors define subexponential size as $\cap_{\varepsilon>0}\mathsf{DTIME}(2^{n^\varepsilon})$, instead of $2^{o(n)}$. Here, the "subexponential" means $\cap_{\varepsilon>0}\mathsf{DTIME}(2^{n^\varepsilon})$.

10.3 Hierarchy: A Fresh Perspective

10.3.1 TM Hierarchy

Theorem 10.3.1: Time Hierarchy Theorem [HS65]

If f and g are time-constructible functions satisfying $f(n)\log f(n) = o(g(n))$, then

$$\mathsf{DTIME}(f(n)) \subsetneq \mathsf{DTIME}(g(n))$$

Proof. (The idea is to use the diagnization technique on a language involving simulating a universal Turing machine for $f(n)$ steps. It can be viewed as a scale-down version of the proof for HALT is undecidable.

For more details, refer to this lecture and P62 on Arora&Barak) ■

Theorem 10.3.1 (Non-Deterministic Time Hierarchy Theorem [Coo73])**.** If f and g are time-constructible functions satisfying $f(n+1) = o(g(n))$, then

$$\mathsf{NTIME}(f(n)) \subsetneq \mathsf{NTIME}(g(n))$$

◊

Proof. (to be done. P63 on Arora&Barak) ■

The following theorem is the space analogue to Theorem 10.3.1. Note that it does not have the $f(n)$ factor that appears in Theorem 10.3.1. This is essentially due to the fact that the universal Turing machine consumes only $S(n)$ space to simulate a $S(n)$-space machine.thm:space-hierarchy

Theorem 10.3.2: Space Hierarchy Theorem [SHL65]

If f, g are space-constructible functions satisfying $f(n) = o\big(g(n)\big)$, then

$$\mathsf{SPACE}\big(f(n)\big) \subsetneq \mathsf{SPACE}\big(g(n)\big)$$

Theorem 10.3.2 (Collapse of PH)**.** PH has the following properties of collapse:

(1) For every $i \geq 1$, $\sum_i^{\mathsf{P}} = \prod_i^{\mathsf{P}}$ implies $\mathsf{PH} = \sum_i^{\mathsf{P}}$.

(2) $\mathsf{P} = \mathsf{NP}$ implies $\mathsf{PH} = \mathsf{P}$.

◊

Proof. We only need to prove the second item. The same argument extends to the first item $i > 1$.

Assume $\mathsf{P} = \mathsf{NP}$, we prove $\mathsf{P} = \mathsf{PH}$ by induction on i that $\sum_1^{\mathsf{P}} \subseteq \mathsf{P}$, which also implies $\prod_i^{\mathsf{P}} \subseteq \mathsf{P}$ as P is closed under complementation.

Assume $\sum_{i-1}^{\mathsf{P}} \subseteq \mathsf{P}$. For any $L \in \sum_i^{\mathsf{P}}$, we immediately have

$$(x, u_1) \in L' \Rightarrow L' \subseteq \textstyle\prod_{i-1}^{\mathsf{P}} \Rightarrow L' \in \mathsf{P},$$

where u_1 is the variable quantified by the first quantifier (the first $\exists$). This means there exists a TM M' the decides L' in polynomial time. Also, by the definition of L and L', it is easy to see that

$$x \in L \Leftrightarrow \exists u_1 \text{ s.t. } (x, u_1) \in L'$$

Then plugging M' into the RHS of the above gives:

$$x \in L \Leftrightarrow \exists u_1 \text{ s.t. } M'(x, u_1) = 1,$$

which means $L \in \mathsf{NP}$. Combining it with our assumption $\mathsf{NP} = \mathsf{P}$, we have $L \in \mathsf{P}$. Since our choice of $L \in \sum_i^{\mathsf{p}}$ is arbitrary, we thus proved $\sum_i^{\mathsf{p}} \subseteq \mathsf{P}$, which finishes our induction step. ■

10.3.2 Circuit Hierarchy and Hard Functions

Theorem 10.3.3 (Existence of hard functions [Sha49])**.** For every $n > 1$, there exists a function $f : \{0,1\}^n \to \{0,1\}$ that cannot be computed by a circuit Cir of size $2n/(10n)$. ◊

Proof. (See Theorem 6.21 in [AB09].) ■

Similar to the time/space hierarchy theorems (Theorem 10.3.1 and 10.3.2), circuits also have a hierarchy theorem.

Theorem 10.3.4 (Non-Uniform Hierarchy Theorem)**.** For every functions $T, T' : \mathbb{N} \to \mathbb{N}$ with $2n/n > T'(n) > 10T(n) > n$,

$$\mathbf{SIZE}(T(n)) \subsetneq \mathbf{SIZE}(T'(n))$$

◊

Proof. (See Theorem 6.22 in [AB09].) ■

10.4 Complete Languages: NP, PSPACE, NL and PH

Around 1971, Cook and Levin independently discovered the notion of NP-completeness and gave examples of combinatorial NP-complete problems whose definition seems to have nothing to do with Turing machines. Soon after, Karp [Kar72] showed that NP-completeness occurs widely and many problems of practical interest are NP-complete, and studied the relations among those problems by Karp reduction.

Theorem 10.4.1: Cook-Levin Theorem [Coo71, Lev73]

Denote by SAT the language of all satisfiable CNF formulae and by 3-SAT the language of all satisfiable 3-CNF formulae. Then

- SAT is NP-complete.
- 3-$\mathsf{SAT} \leq_{\mathsf{p}} \mathsf{SAT}$.

Proof. SAT is obviously in NP. So we only need to prove it is NP-hard by showing $L \leq_{\mathsf{p}}$ SAT for any $L \in$ NP. There are 3 typical ways to do this:

(1) Sipser [Sip12] uses tableau argument.

(2) Arora&Barak [AB09] uses oblivious Turing machine.

(3) Prove CKT-SAT is NP-hard and CKT-SAT $\leq_{\mathsf{p}}$ SAT.

For the first two items, in spite of the difference between the tools use there, these two methods share the same idea of using locality to verify the computation of TMs.

The proof for 3-SAT $\leq_{\mathsf{p}}$ SAT can be done by showing that each clause in a CNF can be broken into small segments of 3-variable clauses, by introducing a new variable for each "breaking" operation. For example, consider a 4-variable clause $C = u_1 \vee u_2 \vee v_3 \vee v_4$. One can easily verify the following is true:

$$C \text{ is satisfiable} \quad \Leftrightarrow \quad (u_1 \vee u_2 \vee z) \wedge (\overline{z} \vee v_3 \vee v_4) \text{ is satisfiable}$$

Applying this (poly-time) transformation on each clause of a CNF gives a 3-CNF, finishing the proof. ■

The first language shown to be PSPACE-complete is TQBF. This is a work of Stockmeyer and Meyer [SM73].

Theorem 10.4.1 (PSPACE-Complete Language [SM73])**.** TQBF is PSPACE-complete, where TQBF denotes the set of quantified Boolean formulae that are true. ◊

Proof. To prove that TQBF is in PSPACE, simply design a recursive algorithm (recursive on the quantifiers) to evaluation the formula. Since space can be reused and for the feedback of each level of recursion (the value need to be stored) is just a single bit, all the work can be done in poly space.

To prove that $L \leq_{\mathsf{p}}$ TQBF for all $L \in$ PSPACE, the main idea is to construct a Boolean formula on the configuration graph of the decider machine for L. (Add details. Check P77 Arora&Barak...) ■

Theorem 10.4.2 (NL-Complete Language)**.** Denote the language PATH as

$$\mathsf{PATH} = \{(G, s, t) \mid \text{ vertex } t \text{ can be reached from } s \text{ in the directed graph } G\ \},$$

Then the following holds:

- PATH is NL-complete
- $\overline{\mathsf{PATH}}$ is NL-complete. (Immerman-Szelepcsényi Theorem [Imm88, Sze87])

◊

Before give the proof of Theorem 10.4.2, we remark that the proof actually can be modified to give the following more general (and surprising) result:

Corollary 10.4.3 (Complete-Equivalence of NSPACE)**.** For every space constructible $f(n) > \log n$, $\mathsf{NSPACE}(f(n)) = \mathsf{coNSPACE}(f(n))$. In particular, $\mathsf{NL} = \mathsf{coNL}$. ◊

Proof for Theorem 10.4.2. As one would expect, the proof uses configuration graph of TM again. (add details according to P80 and P82 of Arora&Barak ...) ∎

We now discuss the case of PH. The following two theorem show an interesting facts: while each $\sum_i^{\mathsf{p}}$ does have its own complete language, the class PH does not, unless PH collapses.

Theorem 10.4.4 (Complete-Collapse of PH)**.** If there exists a language L that is PH-complete, then there exists an i such that $\mathsf{PH} = \sum_i^{\mathsf{p}}$. ◊

Proof. The proof is obvious. ∎

Theorem 10.4.5 (Complete Language for $\sum_i^{\mathsf{p}}$)**.** $\sum_i^{\mathsf{p}}\mathsf{SAT}$ is $\sum_i^{\mathsf{p}}$-complete, where $\sum_i^{\mathsf{p}}\mathsf{SAT}$ denotes the following special version of TQBF problem:

$$\textstyle\sum_i^{\mathsf{p}}\mathsf{SAT} = \{\phi \mid \exists u_1, \forall u_2, \dots, Q_i u_i \quad \phi(u_1, \dots, u_i) = 1\},$$

where ϕ is a Boolean formula, each u_i is a vector of Boolean variables, and Q_i is $\forall$ or $\exists$ depending on whether i is even or odd respectively. ◊

10.4.1 Important Languages and Their Implications

- ANE3SAT, 3SAT, CKTSAT, 3-COL
- TAUTOLOGY, GNI, PRIMALITY, IND-SAT, Linear Programing, PRIMES, Factoring (decisional version): See this lecture.
- ST-PATH: for undirected version, Omer Reingold showed a $\log(n)$-space solution; for the undirected version, it is currently unknown whether $\log(n)$-space solutions are possible. See this lecture. Since we know that ST-PATH can be solved in $O(\log(n))$-space (see this vedio). Also, this problem is NL complete—it can be solved in non-deterministic $O(\log(n))$-space. This also implies that it can be solved in $O(\log^2(n))$-space, by Theorem 10.5.1.
- Here are some P-complete language w.r.t. log-space reduction: HornSAT, Linear Programming, Circuit Evaluation. They have the following implications:
  - $C \in L \Leftrightarrow P \subseteq L$
  - $C \in NL \Leftrightarrow P \subseteq NL$
  - $C \in NC \Leftrightarrow P \subseteq NC$
- ExactClique, SmallestCirccuit: it is unclear whether we can put these two languages in NP or coNP. But it is easy to see that ExactClique is in $\sum_2^{\mathsf{p}}$ and SmallestCircuit is in $\prod_2^{\mathsf{p}}$. See this lecture.

10.5 Time-Space Trade off

You can trade the size of the TM for its efficiency.

Theorem 10.5.1 (Speed-Up Theorem [HS65])**.** if a function f is computable by a TM M in time $T(n)$ then for every constant $c \geq 1$, f is computable by a TM M' in time $T(n)/c$, where M' possibly has larger state size and alphabet size than M. ◊

Remark 10.5.2. Note that [HS65] is a very important paper. It proved the first (but a little relaxed) version of the existence of efficient Universal Turing machine (Theorem 10.1.7), the above speed-up theorem, and the time-hierarchy theorem for deterministic computation. Interestingly, it seems to be this paper that starts the use of the term "computation complexity".

For the trade off between time complexity and space complexity, the following theorem may represents the only non-trial result we currently have. This is rather unsatisfactory since this theorem is not surprising at all.

Theorem 10.5.3. $\mathsf{NSPACE}(S(n)) \subseteq \mathsf{DTIME}(2^{O(S(n))})$ ◊

Proof. (The proof use the configuration graph of Turing machines. P72 on Arora&Barak.)

(This should be simple. Do it soon...) ■

The following theorem reveals the relation between non-deterministic and deterministic power w.r.t. space complexity. It follows an immediately corollary that $\mathsf{NPSACE} = \mathsf{PSPACE}$.

Theorem 10.5.1: Savitch's Theorem [Sav70]

For any space-constructible function $f : \mathbb{N} \to \mathbb{N}$ with $f(n) \geq \log(n)$, the following holds

$$\mathsf{NSPACE}\big(f(n)\big) = \mathsf{PSPACE}\big((f(n))^2\big)$$

Proof. The proof for this theorem follows closely that of 10.4.1. (See P78 of Arora&Barak...) ■

(talk about the trade-off for SAT. P90 of Arora&Barak)

The following theorem reveals that space is a more precious resource than time. This is because, together with space hierarchy theorem (Theorem 10.3.2), it implies that $\mathsf{TIME}\big(t(n)\big) \subsetneq \mathsf{SPACE}\big(t(n)\big)$.

Theorem 10.5.2: [HPV77]

$\mathsf{TIME}\big(t(n)\big) \subseteq \mathsf{SPACE}\big(\frac{t(n)}{\log(t(n))}\big)$

Proof. See this lecture. ■

10.6 Relations among Complexity Classes

(**NL vs L.** ST-PATH is NL complete. Meanwhile, it can be solved in $\log^2(n)$ space. By space hierarchy theorem (Theorem 10.3.2), L is a proper subset of NL.

Theorem 10.6.1: L, NL and P

L $\subset$ P, and NL $\subset$ P.

Proof. To prove that L $\subset$ P: there are only $2^{O(\log n)}$ many different configurations of a decider for a language in L. These can be brute-forced in polynomial time.

To prove that NL $\subset$ P: writing down all the $2^{O(\log n)}$ possible configurations of a non-deterministic log space machine. We want to know whether there is a path form the starting config to the accepting config. This is exactly the PATH problem, which can be sovled in polynomial time. ■

Is L or NL a proper subset of P? If the answer is unknown, what is the critical language (known in NL or L, but not known in P)?

)

Theorem 10.6.2: Levin Recution

If $\mathsf{P} = \mathsf{NP}$, then for every $L \in \mathsf{NP}$ and every $x \in L$, there exists a polynomial-time TM M such that $R_L(x, M(x)) = 1$.

Proof. The proof consists of two steps:

(1) Show such a machine for SAT.

(2) For any $L \in \mathsf{NP}$, show a Levin reduction to SAT.

For the 2nd item, we note that the reduction used in the proof of Cook-Levin theorem (Theorem 10.4.1) is already a Levin reduction. So we only need to do the 1st item.

Assume we have a decider for a SAT. We can then do the following for each variables in order: for the i-th variable v_i, test whether ϕ is still satisfiable with assignment $v_i = 1$ and $v_i = 0$ to figure out the correct value for v_i. If there are n variables in total, such test costs $2n$ calls to the assumed SAT decider, thus can be done in poly time. ■

Theorem 10.6.1. $\mathsf{P} = \mathsf{NP} \Rightarrow \mathsf{EXP} = \mathsf{NEXP}$ ◊

Proof. Hint: use "padding" argument. ■

The following result, Ladner's theorem, shows a surprising fact: if $\mathsf{P} \neq \mathsf{NP}$, there must be some language lying in between P and NP-complete languages.

Theorem 10.6.3: Ladner's theorem—NP intermediate languages [Lad75]

Suppose that $\mathsf{P} \neq \mathsf{NP}$. Then there exists a language $L \in \mathsf{NP} \setminus \mathsf{P}$ that is not NP-complete.

Proof. (to be done. P64 on Arora&Barak) ■

Theorem 10.6.4: Manhany's Theorem

if any "sparse language" is NP-Complete, then $P = NP$.

Proof. See this lecture. ■

Theorem 10.6.2. $\mathsf{PH} \subseteq \mathsf{PSPACE}$. Morover, if $\mathsf{PH} \subsetneq \mathsf{PSPACE}$, PH collapses to $\sum_i^{\mathsf{p}}$ for some i. ◊

Proof. The first part is obvious. The second part follows as a simple corollary of Theorem 10.4.4 plus Theorem 10.4.1. ■

We next show a seemingly straightforward result. However, its proof is non-trivial and gives another approach to prove Cook-Levin Theorem (Theorem 10.4.1). We give more details in Remark 10.6.4

Theorem 10.6.3. $\mathsf{P} \subsetneq \mathsf{P/poly}$. ◊

Proof. This proof uses oblivious Turing machine. There can at most be polynomially many snapshots of a poly time oblivious Turing machine. And the transition between each two adjacent snapshots can be verified by a constant size circuit due to the locality of such TM. Thus, in total, the computation can be done by a poly size circuit that sequentially verifies the adjacent snapshots.

To see this subset relation is proper, consider the unary halting problem. ■

Remark 10.6.4. The idea of the proof already implies that $\mathsf{CKT\text{-}SAT}$ is $\mathsf{P/poly}$-hard. If we can show $\mathsf{CKT\text{-}SAT} \leq_{\mathsf{p}} \mathsf{3\text{-}SAT}$, we then have another proof for Cook-Levin theorem. Actually, converting a circuit to 3-CNF is easy with the following rules to convert each type of gate:

- AND Gate: $z_1 = z_2 \wedge z_3 \quad \Leftrightarrow \quad (\overline{z}_1 \vee \overline{z}_2 \vee z_3) \wedge (\overline{z}_1 \vee z_2 \vee \overline{z}_3) \wedge (\overline{z}_1 \vee z_2 \vee z_3) \wedge (z_1 \vee \overline{z}_2 \vee \overline{z}_3)$
- OR Gate: $z_1 = (z_2 \vee z_3) \quad \Leftrightarrow \quad (z_1 \vee \overline{z}_2) \wedge (z_1 \vee \overline{z}_3) \wedge (\overline{z}_1 \vee z_2 \vee z_3)$
- NOT Gate: $z_1 = \overline{z}_2 \quad \Leftrightarrow \quad (z_1 \vee z_2) \wedge (\overline{z}_1 \vee \overline{z}_2)$
- For output wire y of $\mathsf{CKT\text{-}SAT}$, add (y) as the 3-CNF clause.

We have already seen that $\mathsf{P} \subsetneq \mathsf{P/poly}$ in Theorem 10.6.3. The following result of Karp and Lipton [KL82] provides some evidence for the conjecture that $\mathsf{P} \neq \mathsf{NP}$.

Theorem 10.6.5 (Karp-Lipton Theorem [KL82])**.** $\mathsf{NP} \subseteq \mathsf{P/poly}$ implies $\mathsf{PH} = \sum_2^{\mathsf{p}}$. ◊

Proof. According to Theorem 10.3.2, it is sufficient if we can show that $\mathsf{NP} \subseteq \mathsf{P/poly}$ implies $\prod_2^{\mathsf{p}} \subseteq \sum_2^{\mathsf{p}}$. To do that, it suffices to show $\prod_2^{\mathsf{p}}\mathsf{SAT} \in \sum_2^{\mathsf{p}}$.

Recall that

$$\phi \in \textstyle\prod_2^{\mathsf{p}}\mathsf{SAT} \quad \Leftrightarrow \quad \forall u_1 \in \{0,1\}^n, \exists u_2 \in \{0,1\}^n \quad \phi(u_1, u_2) = 1. \tag{10.1}$$

Define the following language for tuples of Boolean formula ϕ and a variable $u_1 \in \{0,1\}^n$:

$$\mathcal{L}_1 = \{(\phi, u_1) \mid \exists u_2 \in \{0,1\}^n \text{ s.t. } \phi(u_1, u_2) = 1\}$$

Obviously, $\mathcal{L}_1 \in \mathsf{NP}$. Since we assume $\mathsf{NP} \subseteq \mathsf{P/poly}$, there exists a poly-size circuit family $\{C_n\}_{n\in\mathbb{N}}$ deciding $\mathcal{L}_1$. In another word, $\{C_n\}_{n\in\mathbb{N}}$ is a (family of) decider for the SAT problem of formulea of the form $\phi(u_1, \cdot)$.

Recall that in the proof of Theorem 10.6.2, we showed how to efficiently construct a witness extractor from the decider of SAT. Thus, there exists a poly-size circuit family $\{C'_n\}_{n\in\mathbb{N}}$ such that $\phi\big(u_1, C'_n(\phi, u_1)\big) = 1$ if (ϕ, u_1) is in $\mathcal{L}_1$ (i.e. $\phi(u_1, \cdot)$ is satisfiable). Note that if $\{C'_n\}_{n\in\mathbb{N}}$ is of size $p(n)$, we can describe C'_n using a binary string of size $q(n) = \mathsf{poly}\big(p(n)\big)$, which is also some polynomial on n.

We then denote the following language:

$$\phi \in \mathcal{L}_2 \quad \Leftrightarrow \quad \exists w \in \{0,1\}^{q(n)}, \forall u_1 \in \{0,1\}^n \quad M'(\phi, u_1, w) = 1 \tag{10.2}$$

where M' is a TM such that on input (ϕ, u_1, w), it interprets w as the description of a C'_n and outputs 1 iff $\phi\big(u_1, C'_n(\phi, u_1)\big) = 1$.

Obviously, $\mathcal{L}_2 \in \sum_2^{\mathsf{p}}$. Morover, with a little thinking, one can see that the expressions (10.1) and (10.2) essentially desribe the same language, i.e. $\mathcal{L}_2 = \sum_2^{\mathsf{p}}\mathsf{SAT}$. Thus, $\prod_2^{\mathsf{p}}\mathsf{SAT} \in \sum_2^{\mathsf{p}}$. This closes our proof. ■

A similar result to Theorem 10.6.5 (also appeared in [KL82], but was attributed to Meyer) can be proved for EXP.

Theorem 10.6.6 (Meyer's Theorem [KL82])**.** $\mathsf{EXP} \subseteq \mathsf{P}_{/\mathsf{poly}}$ implies $\mathsf{EXP} = \sum_2^{\mathsf{p}}$. In particular, $\mathsf{EXP} \subseteq \mathsf{P}_{/\mathsf{poly}}$ implies $\mathsf{P} \neq \mathsf{NP}$. ◊

Proof. (It again uses oblivious Turing and snapshots. Do it later ... P102 Arora&Barak.) ■

Chapter 11

Proof Systems: Bridging Crypto and Complexity Theory

11.1 PCP Theorem

Theorem 11.1.1: The PCP Theorem

$\mathsf{NP} \subseteq \mathsf{PCP}_{1,\frac{1}{2}}[O(\log n), O(1)]$.

11.1.1 Exponential-Size PCP

Linear PCP. The first step is to construct a *linear PCP* (LPCP) for the (NP complete) language of quadratic equation satisfiability. More concretely, one can show that $\mathsf{QUAD\text{-}EQ} \in \mathsf{LPCP}_{1,\frac{3}{4}}[n^2, O(n+m), 4]$, where n is the number of variables and m is the number of equations. The core idea in this proof is the design of a tensor product. Namely, if the verifier is given a oracle claimed to be a tensor product $x \otimes x$ between the same vector, it should be able to check it.

Exponential-Size PCP for NP. LPCP guarantees that once the verifier gets access to linear oracle, soundness is guaranteed. To build the final PCP by employing the LPCP verifier, we now need to enforce the prover to use a linear oracle. Put it in another way, we need to develop a method for the verifier for linearity testing: if the prover gives a non-linear function as the oracle, the verifier must catch it with good probability. Assume we have such a linearity test, then the PCP can be built in the following way:

- The verifier performs the linearity test [BLR90] to check that the oracle is really a linear function.
- If the last check passes, the verifier can safely assume that the oracle is a real linear function. It then simply runs the LPCP verifier;

The soundness $\varepsilon_{\mathsf{PCP}}$ of this PCP is upper bounded by $\max\{\varepsilon_{\mathsf{LPCP}}, \varepsilon_{\mathsf{LIN}}\}$, where $\varepsilon_{\mathsf{LIN}}$ is the soundness error of the linearity test.

However, the are two caveat when we implement the above idea:

1. Ideally, we want to have a linearity test such that if the oracle is not a linear function, the verifier accepts with probability $< \varepsilon_{\mathsf{LIN}}$. But this is impossible unless the verifier checks the whole true table of the function. For example, we can have a non-linear function that differs

with some linear function on only a single input. Thus, to ensure the verifiers' efficiency, we have to tolerate some slackness. The linearity test we will have can only guarantee that: if a function is far (say 10%-far, in terms of hamming distance of the truth table) from any linear function, the verifier can be fooled with probability $< \varepsilon_{\mathsf{LIN}}$.[1] This introduces some "middle land" to the above soundness analysis: a malicious prover can generate a non-linear function that is only $< 10\%$-far from some linear function. Nevertheless, this middle-land case can be handled by union bound, introducing only a $q \cdot \frac{1}{10}$ additional error term to $\varepsilon_{\mathsf{PCP}}$, where q is the number of the verifier's queries in the whole execution. Rigorously, assuming that there is a linear function $\langle \alpha, \cdot \rangle$ that is 10%-close to the maliciously generated oracle $\widetilde{\pi}$, when we run the LPCP verifier w.r.t. $\widetilde{\pi}$, we have:

$$\begin{aligned}
\Pr\left[V_{\mathsf{LPCP}}^{\widetilde{\pi}}(x) = 1\right] &\le \Pr\left[V_{\mathsf{LPCP}}^{\widetilde{\pi}}(x) = 1 \;\middle|\; \begin{array}{l}\text{All queries land in}\\ \text{good locations}\end{array}\right] + \Pr\left[\begin{array}{l}\text{At least one } \mathsf{LPCP} \text{ query}\\ \text{lands in bad locations}\end{array}\right] \\
&= \Pr\left[V_{\mathsf{LPCP}}^{\langle \alpha, \cdot \rangle}(x) = 1 \;\middle|\; \begin{array}{l}\text{All queries land in}\\ \text{good locations}\end{array}\right] + \Pr\left[\begin{array}{l}\text{At least one } \mathsf{LPCP} \text{ query}\\ \text{lands in bad locations}\end{array}\right] \\
&\le \varepsilon_{\mathsf{LPCP}} + q \cdot \frac{1}{10} \qquad (11.1)
\end{aligned}$$

However, the above bound is not informative if $q \cdot \frac{1}{10}$ is close (or equal) to 1. This can be fixed by repetition: if we repeat each query of the underlying LPCP t times (with fresh randomness), we can drive the term in Inequality (11.1) down to $\varepsilon_{\mathsf{LPCP}} + q \cdot \frac{1}{10^t}$.

2. The second caveat is about Inequality (11.1). The $\frac{1}{10}$ terms is due to the (ideal-case) fact that the (linear-PCP) verifier's query is uniformly distributed over all the positions. However, in the underlying linear PCP, the verifier's query is not uniformly. But this can be fixed by *self-correction*: every time the V_{LPCP} want to query a position z, it samples a r uniformly at random, and queries both position r and $z + r$. Then it computes $\pi(z) = \pi(r) + \pi(r + z)$ (due to the linearity of the linear PCP π). Now we can safely say that both r and $r + z$ are random queries. However, note that soundness is broken if one of these two queries lands in bad locations, which happens with probability $\frac{2}{10}$ (if $\frac{1}{10}$ of the oracle is bad). Thus, with this self-correction (and the aforementioned repetition), the term in Inequality (11.1) should be $\varepsilon_{\mathsf{LPCP}} + q \cdot \frac{2}{10^t}$.

Remark 11.1.2: On the BLR Linearity Test

The BLR test is very simple: to test the linearity of a function f on $\{0,1\}^n$, just pick $x, y \xleftarrow{\$} \{0,1\}^n$ and check if $f(x) + f(y) = f(x + y)$. All the hard work lies in its soundness analysis. The original [BLR90] paper showed that $\Pr\left[V_{\mathsf{LIN}}^{f} = 0\right] \ge \min\{\frac{2}{9}, \frac{\delta(f)}{2}\}$, where δf is the fractional hamming distance between f and the closest linear function to it. Their proof used only elementary probability theory argument in an elegant way. Later, relying on tools from Boolean Fourier Analysis, [BCH$^+$95] improved the soundness analysis of BLR test by showing that $\Pr\left[V_{\mathsf{LIN}}^{f} = 0\right] \ge \delta(f)$. Put it in another way, if we use BLR test with soundness error $\varepsilon_{\mathsf{LIN}}$, we can make sure that the target function is $\varepsilon_{\mathsf{LIN}}$-close to some linear function.

[1] For the linearity test we will use, this error is actually the inverse of the "slackness" factor: if a function is $\varepsilon_{\mathsf{LIN}}$-far from linear, the verifier can be fooled with with probability $< \varepsilon_{\mathsf{LIN}}$. See Remark 11.1.2.

On the Soundness Error. The above analysis says that: for $\widetilde{\pi}$ of a false statement $x \notin \mathcal{L}$,

- if $\widetilde{\pi}$ is $\varepsilon_{\mathsf{LIN}}$-far from any linear function, V will accept (mistakenly) with probability $\leq \varepsilon_{\mathsf{LIN}}$ (due to Remark 11.1.2); (note that here we only consider the soundness error of the linearity test. This is because if $\widetilde{\pi}$ manages to pass the linearity test, the LPCP test is not reliable at all. To the extreme, if $\widetilde{\pi}$ is $\varepsilon_{\mathsf{LIN}}$-far from any linear function, the $V^{\widetilde{\pi}}_{\mathsf{LPCP}}(x)$ may accept with probability 1.)

- if $\widetilde{\pi}$ is $\varepsilon_{\mathsf{LIN}}$-close from some linear function (the linearity test is not reliable), the above analysis tells us that V will accept (mistakenly) with probability $\leq \varepsilon_{\mathsf{LPCP}} + q \cdot 2 \cdot \varepsilon^t_{\mathsf{LIN}}$.

Thus, the soundness error of the above PCP is $\varepsilon_{\mathsf{PCP}} \leq \max\{\varepsilon_{\mathsf{LIN}}, \varepsilon_{\mathsf{LPCP}} + q \cdot 2 \cdot \varepsilon^t_{\mathsf{LIN}}\}$.

On the Complexity. The above analysis is a generic compiler from LPCP to PCP. It shows that

$$\mathsf{LPCP}_{1,\epsilon_{\mathsf{LPCP}}}[\ell, r, q] \subseteq \mathsf{PCP}_{1,\epsilon_{\mathsf{PCP}}}[2^\ell, 2\ell + 2qt\ell, 3 + qt], \text{ where } \varepsilon_{\mathsf{PCP}} \leq \max\{\varepsilon_{\mathsf{LIN}}, \varepsilon_{\mathsf{LPCP}} + q \cdot 2 \cdot \varepsilon^t_{\mathsf{LIN}}\}$$

We can set $\varepsilon_{\mathsf{LIN}} = \frac{1}{2}$ and set $t = O(\log q)$ such that $q \cdot 2 \cdot \varepsilon^t_{\mathsf{LIN}}$ is an arbitrarily small constant, e.g. $\frac{1}{100}$. Since we know that QUAD-EQ $\in \mathsf{LPCP}_{1,\frac{1}{2}}[\mathsf{poly}(n), \mathsf{poly}(n), O(1)]$, the above parameter setting gives us that:

$$\mathsf{NP} \subseteq \mathsf{PCP}_{1,\frac{1}{2}+\frac{1}{100}}[\exp(n), \mathsf{poly}(n), O(1)].$$

As a historical remark, this exponential size PCP with constant number of queries is the inner PCP in the work [ALM+92, ALM+98].

11.1.2 Polynomial-Size PCP

11.2 PCP of Proximity

Resources

- The first work that formalized PCPP was [BGH+04], where PCPP was defined w.r.t. pair languages. It also contains a discussion about pair languages vs standard languages, and PCP vs PCPP.
- [DK12] contains the same formalism as in [BGH+04]. It indicates that in the [BGH+04] construction, the PCPP proof oracle can be constructed efficiently.
- Section A.5.1 of [GOSV14] also contains a clean formalism of PCPP. It basically summarized the things in [BGH+04] and [DK12].
- [IW14] has a informal description of PCPP w.r.t. standard NP languages. This is the only definition I found that did not use pair languages.

11.3 Interactive Proofs and Arthur-Merlin Games

(A good starting point for this topic is the introduction of [GS86].)

11.3.1 Multi-Prover Interactive Proofs

Chapter 12

Cryptographic Reductions and Impossibility Results

Some Comments on Reductions

It is worth noting that relativizing constructions generalize *fully-black-box* constructions in the sense that every fully-black-box construction is also a relativizing one [RTV04]. Thus, impossibility results for relativizing constructions are stronger. Indeed, several works (e.g.,[IR89, Sim98, GKM$^+$00]) proved impossibility results for fully-black-box constructions by ruling out relativizing constructions. There are also impossibility results for fully-black-box reductions *without* ruling out relativizing reductions, e.g., [HR04, Haj18].
We also remark that the relativizing reductions as defined in [RTV04] require that the adversary is a PPT oracle machine. Alternatively, one can define relativizing reduction by allowing the adversary to be inefficient, as long as it only makes polynomially many oracle queries. (The construction should of course be efficient). This will not affect the impossibility result as fully-black-box reductions also imply this version of relativizing reduction.
Another paradigm appeared in [GGKT05] (the merged version of [GT00, GGK03]). Their approach can be demonstrated by their separation of "efficient" PRG from TDP. To show that, they argue in the following way: Given a length-preserving random oracle, if an "efficient" PRG can be constructed, then $\mathsf{P} \neq \mathsf{NP}$. They argued that such a result ruled out the **weak-black-box** reduction from "efficient" PRG to TDP, which is a stronger impossibility result than fully/semi black-box separation. I have two questions:

- How to argue formally that their claim rules out the weak black-box separation?
- Is it true that they even rule out some reduction that is more general than weak black-box reduction. Is there a formal definition/name for this more-general reduction?
- Is it true that Barak's non-black-box ZK uses a weak-black-box reduction, but not a semi-black-box reduction?

12.1 Polynomial-Time Reductions with Expected Polynomial-Time Adversaries

The Problem. To prove the security of cryptographic objects, we usually need to conduct polynomial time reductions. For example, consider a primitive A that is constructed (solely) from another primitive B which is secure by assumption (for concreteness, think of A as a commitment scheme and B as an one-way permutation). To prove that A achieves some desired security property, the canonical approach is to assume, for the sake of contradiction, that there exists a probabilistic polynomial-time (PPT) adversary Adv, which is able to break the target security property of A. Then, the proof is done if one can show an efficient way (i.e. the "reduction") to make use of Adv to break the security of B.

Everything is good if the security of B holds against the class of adversaries for which we try to prove the security of A. For example, assume that B is secure against all PPT adversaries. Then the security of A against all PPT adversaries will be established once we can finish the reduction in PPT. This type of arguments extends to other class of adversaries. For example, similar results hold for the class of sub-exponential adversaries, with the reduction being sub-exponential time.

But things become a little bit tricky if we want to prove the security of A against a class of adversaries which is stronger than the ones that are ruled out by the security of B. This would not be a reasonable concern if the power of the two classes of adversaries differs too much. Indeed, no one will hope to construct sub-exponentially secure commitments from one-way permutations that are only polynomially secure. However, this difference on power could sometimes be so small that we have to pay attention. One such setting is:

> B is secure against PPT adversaries, but we want to prove the security of A against *expected* PPT adversaries.

The above setting appeared in the literature quite often, e.g., Claim 3 in [GK96], Lemma 4.3 in [BJY97], and Proposition 3.3.30 in Feige's thesis [Fei90][1].

The trick is to employ an interesting combination of averaging argument and Markov's inequality. Though the papers mentioned above are already very well-written with enough details, there are still several steps that may not be straightforward to a beginner. Thus, I decided to take a note here to present a thorough derivation.

The Solution: truncating the execution. To illustrate the core of the this technique, I will show the following lemma about OWFs (though all the aforementioned papers are about zero-knowledge). One can easily extend this argument to proper contexts as he/she needs.

Lemma 12.1.1:

Assume f is an OWF against PPT adversaries. Then it is also an OWF against expected PPT adversaries.

Proof. In this proof, I will gloss over the details that can be inferred from the context easily, such as the length of the pre-images/images of f.

Assume, for the sake of contradiction, that there is an expected PPT machine Adv that breaks the one-wayness of f. We will build a machine Adv' that runs in strictly polynomial time and (still) breaks the one-wayness of f.

Let λ denote the security parameter. W.l.o.g., assume that the expected running time of Adv is the polynomial $T(\lambda)$. It breaks the one-wayness of f, which means that there exist a polynomial $P(\lambda)$ such that for infinitely many $\lambda \in \mathbb{N}$, Adv inverts f with probability at least $\frac{1}{P(\lambda)}$. In the remaining part of this section, I will drop λ from $T(\lambda)$ and $P(\lambda)$ to make the presentation succinct.

[1] Noticeably, [Fei90] devoted the whole Chapter 3 to this issue, presenting a beautiful and thorough discussion.

The machine Adv' is constructed by "truncating" the executions of Adv that go beyond $2TP$ steps. In the following, we argue that Adv' also breaks the one-wayness of f.

First, it follows from Markov's inequality that the truncated executions count for only a small portion. More formally, let random variable X denote the running time of Adv. According Markov's inequality, we have:

$$\Pr[X \geq 2TP] \leq \frac{1}{2P}.$$

Then, by an averaging argument, one can show that in the "un-truncated" executions, Adv (the de facto Adv') can still invert f with probability at least $\frac{1}{2P}$. Formally, let the $\mathsf{Win}(\lambda)$ denote the even that Adv wins in the security game for the one-wayness of f, with security parameter set to λ. We then have:

$$\Pr[\mathsf{Win}] = \Pr\left[\mathsf{Win}|X < 2TP\right] \cdot \Pr\left[X < 2TP\right] + \Pr\left[\mathsf{Win}|X \geq 2TP\right] \cdot \Pr\left[X \geq 2TP\right]$$

We now prove that the above equation implies that

$$\Pr\left[\mathsf{Win}|X < 2TP\right] \geq \frac{1}{2P}. \tag{12.1}$$

Assume for contradiction that $\Pr\left[\mathsf{Win}|X < 2TP\right] < \frac{1}{2P}$. Continuing the above equation with this assumption, we have:

$$\begin{aligned}\frac{1}{P} = \Pr[\mathsf{Win}] &< \frac{1}{2P} \cdot \Pr\left[X < 2TP\right] + \Pr\left[\mathsf{Win}|X \geq 2TP\right] \cdot \frac{1}{2P} \\ &\leq \frac{1}{2P} \cdot 1 + 1 \cdot \frac{1}{2P} \\ &= \frac{1}{P},\end{aligned}$$

which implies a contradiction as it says $\frac{1}{P} < \frac{1}{P}$. Thus, Equation (12.1) holds. This finishes our proof as $\Pr\left[\mathsf{Win}|X < 2TP\right]$ is exactly the probability that Adv' wins in the security game. ■

When not to use the truncating argument. I want to highlight a restriction of the above "truncating" argument: the winning probability of Adv will drop by a constant fraction. In the above example, the winning probability of Adv is $\geq \frac{1}{P}$, but the winning probability of Adv' can only be shown to be $\geq \frac{1}{2P}$. While this is usually not a problem in crypto reductions, this does restrict its applicability in analyses for ZK simulators or knowledge extractors.

For example, to prove ZK property, we usually construct a simulator Sim that runs in expected polynomial time, and outputs a simulated view indistinguishable from (in other words, negligibly-close to) a real one. One may wondering if it is possible to use the above truncating argument to construct a strictly-polynomial time Sim' that is just as good as Sim. But this does not work because the output of Sim' will not be negligibly-close to that of Sim (due to the constant fraction drop). Therefore, the output of Sim' will not be negligibly-close to the real view anymore.[2]

Some Afterthoughts. The above discussion seems to give us more confidence about the thought

[2]Indeed, this limitation is inherent. [BL02] shows that constant-round zero-knowledge protocols admitting strictly-polynomial-time black-box simulation are possible only for languages in BPP.

that “expected polynomial time” is indeed a reasonable relaxation. This is good news due to the following fact—expected polynomial time is actually inherent in all the fully-black-box reductions (in the terminology of [RTV04]) for zero-knowledge property or arguments/proofs of knowledge, if one insists on constant-round constructions from standard assumptions [BL02].

However, the following another-side view may stop you from celebrating the good news. The above argument seems to say that allowing the adversary to run in expected polynomial time does not make much difference. For example, Lemma 12.1.1 essentially says that requiring one-wayness to hold against all expected PPT adversaries would eventually result in the same definition of OWFs as the standard one. If so, then why are cryptographers trying so hard to distinguish between the strict polynomial time and the expected one?

The short answer is—expected polynomial-time simulation/extraction is not closed under composition. Elaborating on this point could require another long article. For those who are interested, see the introduction section of the insightful work of Barak and Lindell [BL02] and the references there.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009. 28, 29, 57, 67, 68

[ADRS15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete Gaussian sampling: Extended abstract. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 733–742, Portland, OR, USA, June 14–17, 2015. ACM Press. 51

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108, Philadephia, PA, USA, May 22–24, 1996. ACM Press. 52

[Ajt98] Miklós Ajtai. The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract). In *30th Annual ACM Symposium on Theory of Computing*, pages 10–19, Dallas, TX, USA, May 23–26, 1998. ACM Press. 51

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *33rd Annual ACM Symposium on Theory of Computing*, pages 601–610, Crete, Greece, July 6–8, 2001. ACM Press. 51

[ALM$^+$92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, Pittsburgh, PA, USA, October 24–27, 1992. IEEE Computer Society Press. 76

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998. 76

[AP13] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 1–20, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. 55

[AR04] Dorit Aharonov and Oded Regev. Lattice problems in NP cap coNP. In *45th Annual Symposium on Foundations of Computer Science*, pages 362–371, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press. 51

[Axl15] Sheldon Axler. *Linear algebra done right.* Springer, 2015. 19

[BCH$^+$95] Mihir Bellare, Don Coppersmith, Johan Håstad, Marcos A. Kiwi, and Madhu Sudan. Linearity testing in characteristic two. In *36th Annual Symposium on Foundations of Computer Science*, pages 432–441, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press. 75

[BFNW93] Lźszló Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. Bpp has subexponential time simulations unlessexptime has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993. 1

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 335–359, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. 46, 47

[BGH$^+$04] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, June 13–16, 2004. ACM Press. 76

[BJY97] Mihir Bellare, Markus Jakobsson, and Moti Yung. Round-optimal zero-knowledge arguments based on any one-way function. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 280–305, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany. 31, 79

[BL02] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *34th Annual ACM Symposium on Theory of Computing*, pages 484–493, Montréal, Québec, Canada, May 19–21, 2002. ACM Press. 80, 81

[BLMZ18] James Bartusek, Tancrède Lepoint, Fermi Ma, and Mark Zhandry. New techniques for obfuscating conjunctions. Technical report, Cryptology ePrint Archive, Report 2018/936, 2018. https://eprint. iacr. org . . . , 2018. 46

[BLR90] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *22nd Annual ACM Symposium on Theory of Computing*, pages 73–83, Baltimore, MD, USA, May 14–16, 1990. ACM Press. 74, 75

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. 54

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. 41

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. 53

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *32nd Annual ACM Symposium on Theory of Computing*, pages 235–244, Portland, OR, USA, May 21–23, 2000. ACM Press. 1

[CLMP12] Kai-Min Chung, Edward Lui, Mohammad Mahmoody, and Rafael Pass. Unprovable security of two-message zero knowledge. Technical report, Department of Computer Science, Cornell University, Ithaca, NY, 2012. 31

[Coo71] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971. 67

[Coo73] Stephen A Cook. A hierarchy for nondeterministic time complexity. *Journal of Computer and System Sciences*, 7(4):343–353, 1973. 66

[CW79] J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2):143–154, 1979. 43

[Dam88] Ivan Damgård. Collision free hash functions and public key signature schemes. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology – EUROCRYPT'87*, volume 304 of *Lecture Notes in Computer Science*, pages 203–216, Amsterdam, The Netherlands, April 13–15, 1988. Springer, Heidelberg, Germany. 41

[DF04] David Steven Dummit and Richard M Foote. *Abstract algebra*, volume 3. Wiley Hoboken, 2004. 12, 20

[DGH$^+$19] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Daniel Masny, and Daniel Wichs. Two-round oblivious transfer from cdh or lpn. Cryptology ePrint Archive, Report 2019/414, 2019. https://eprint.iacr.org/2019/414. 33

[DGI$^+$19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 54

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, South Korea, February 13–15, 2001. Springer, Heidelberg, Germany. 39

[DK12] Dana Dachman-Soled and Yael Tauman Kalai. Securing circuits against constant-rate tampering. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 533–551, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. 76

[DORS03] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. Cryptology ePrint Archive, Report 2003/235, 2003. https://eprint.iacr.org/2003/235. 46, 47, 48, 49

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. 47, 48, 49

[Fei90] Uriel Feige. *Alternative models for zero knowledge interactive proofs*. PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1990. 79

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany. 41

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. 46

[GG98] Oded Goldreich and Shafi Goldwasser. On the limits of non-approximability of lattice problems. In *30th Annual ACM Symposium on Theory of Computing*, pages 1–9, Dallas, TX, USA, May 23–26, 1998. ACM Press. 51

[GGK03] Rosario Gennaro, Yael Gertner, and Jonathan Katz. Lower bounds on the efficiency of encryption and digital signature schemes. In *35th Annual ACM Symposium on Theory of Computing*, pages 417–425, San Diego, CA, USA, June 9–11, 2003. ACM Press. 78

[GGKT05] Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM journal on Computing*, 35(1):217–246, 2005. 78

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, June 1996. 79

[GKM$^+$00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science*, pages 325–335, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press. 78

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 38

[Gol08] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. 61

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany. 63

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. 63

[GOSV14] Vipul Goyal, Rafail Ostrovsky, Alessandra Scafuro, and Ivan Visconti. Black-box non-black-box zero knowledge. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 515–524, New York, NY, USA, May 31 – June 3, 2014. ACM Press. 76

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press. 52

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *18th Annual ACM Symposium on Theory of Computing*, pages 59–68, Berkeley, CA, USA, May 28–30, 1986. ACM Press. 77

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. 12

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. 53

[GT00] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st Annual Symposium on Foundations of Computer Science*, pages 305–313, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press. 78

[Haj18] Mohammad Hajiabadi. Enhancements are blackbox non-trivial: Impossibility of enhanced trapdoor permutations from standard trapdoor permutations. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 448–475, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. 31, 78

[HHPS11] Shai Halevi, Danny Harnik, Benny Pinkas, and Alexandra Shulman-Peleg. Proofs of ownership in remote storage systems. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 2011: 18th Conference on Computer and Communications Security*, pages 491–500, Chicago, Illinois, USA, October 17–21, 2011. ACM Press. 41, 42

[HL18] Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 850–858, Paris, France, October 7–9, 2018. IEEE Computer Society Press. 41

[HLWW13] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 160–176, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. 46

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963. 35

[HPSS08] Jeffrey Hoffstein, Jill Pipher, Joseph H Silverman, and Joseph H Silverman. *An introduction to mathematical cryptography*, volume 1. Springer, 2008. 51

[HPV77] John Hopcroft, Wolfgang Paul, and Leslie Valiant. On time versus space. *Journal of the ACM (JACM)*, 24(2):332–337, 1977. 70

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 92–105, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. 31, 78

[HR07] Ishay Haviv and Oded Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing*, pages 469–477, San Diego, CA, USA, June 11–13, 2007. ACM Press. 51

[HS65] Juris Hartmanis and Richard E Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965. 66, 70

[HS66] Fred C Hennie and Richard Edwin Stearns. Two-tape simulation of multitape turing machines. *Journal of the ACM (JACM)*, 13(4):533–546, 1966. 61

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, WA, USA, May 15–17, 1989. ACM Press. 46

[Imm88] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on computing*, 17(5):935–938, 1988. 68

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. 1

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, WA, USA, May 15–17, 1989. ACM Press. 78

[IW14] Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 121–145, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. 76

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *15th Annual ACM Symposium on Theory of Computing*, pages 193–206, Boston, MA, USA, April 25–27, 1983. ACM Press. 51

[Kar72] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972. 67

[Kho04] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. In *45th Annual Symposium on Foundations of Computer Science*, pages 126–135, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press. 51

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th Annual ACM Symposium on Theory of Computing*, pages 723–732, Victoria, BC, Canada, May 4–6, 1992. ACM Press. 42

[Kil94] Joe Kilian. On the complexity of bounded-interaction and noninteractive zero-knowledge proofs. In *35th Annual Symposium on Foundations of Computer Science*, pages 466–477, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press. 41

[KK20] Nathan Keller and Ohad Klein. Proof of tomaszewski's conjecture on randomly signed sums, 2020. 33

[KL82] Richard M. Karp and Richard J. Lipton. Turing machines that take advice. *L'Ensignement Mathématique*, 28:191–210, 1982. 72, 73

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014. 8, 9

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science*, pages 364–373, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. 39

[KOS18] Dakshita Khurana, Rafail Ostrovsky, and Akshayaram Srinivasan. Round optimal black-box "commit-and-prove". In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 286–313, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. iii, 16, 58

[Kup05] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005. 1

[KW17] Sam Kim and David J. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 503–536, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 54

[KW19] Sam Kim and David J. Wu. Watermarking PRFs from lattices: Stronger security via extractable PRFs. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 335–366, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 54

[Lad75] Richard E Ladner. On the structure of polynomial time reducibility. *Journal of the ACM (JACM)*, 22(1):155–171, 1975. 71

[Lev73] Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973. 67

[LLL82] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. 51

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. 55

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. 55

[Mic98] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. In *39th Annual Symposium on Foundations of Computer Science*, pages 92–98, Palo Alto, CA, USA, November 8–11, 1998. IEEE Computer Society Press. 51

[Mil01] Peter Bro Miltersen. Derandomizing complexity classes. *COMBINATORIAL OPTIMIZATION-DORDRECHT-*, 9(2):843–941, 2001. 1

[MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 465–484, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany. 53

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. 53

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 21–39, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. 52

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press. 52

[MV10] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In Leonard J. Schulman, editor, *42nd Annual ACM Symposium on Theory of Computing*, pages 351–358, Cambridge, MA, USA, June 5–8, 2010. ACM Press. 51

[NC11] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, USA, 10th edition, 2011. 21, 22, 23, 26, 27

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of computer and System Sciences*, 49(2):149–167, 1994. 28

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, pages 33–43, Seattle, WA, USA, May 15–17, 1989. ACM Press. 41, 43

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996. 48

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of

Lecture Notes in Computer Science, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. 39

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 160–176, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. 1

[Pas04] Rafael Pass. Alternative variants of zero-knowledge proofs. Master's thesis, Royal Institute of Technology, Stockholm, Sweden, 2004. 1

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. 53

[Pei15] Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939, 2015. `https://eprint.iacr.org/2015/939`. 50, 51, 53, 55

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196, Victoria, BC, Canada, May 17–20, 2008. ACM Press. 47

[Reg04] Oded Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. *arXiv preprint quant-ph/0406151*, 2004. 1

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. 53, 54

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 78, 81

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553, New York, NY, USA, October 17–19, 1999. IEEE Computer Society Press. i, 16

[Sav70] Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, 4(2):177–192, 1970. 70

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical computer science*, 53(2-3):201–224, 1987. 51

[Sha49] Claude E Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, 1949. 67

[SHL65] Richard Edwin Stearns, Juris Hartmanis, and Philip M Lewis. Hierarchies of memory limited computations. In *6th Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1965)*, pages 179–190. IEEE, 1965. 66

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999. 51

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany. 78

[Sip12] Michael Sipser. *Introduction to the Theory of Computation*. Cengage learning, 2012. 68

[SM73] LJ Stockmeyer and AR Meyer. Word problems requiring exponential time: preliminary report, in5th symp. on theory of computing'. 1973. 68

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. 17

[Sze87] Róbert Szelepcsényi. The method of forcing for nondeterministic automata. *Bull. European Associ. Theoret. Compur. Sci*, 33:96100, 1987. 68

[Tha20] Justin Thaler. Proofs, arguments, and zero-knowledge, 2020. Lecture Notes for COSC 544 at Georgetown University,. 62

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012. 44

[WC81] Mark N. Wegman and J. Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981. 43, 44

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st Annual Symposium on Foundations of Computer Science*, pages 531–540, Las Vegas, NV, USA, October 23–26, 2010. IEEE Computer Society Press. 2