Leftover Hash Lemma and its Applications

rkm0959 (Gyumin Roh)

October 18th

Outline

Introduction and Motivation

Bunch of Terminology

The Leftover Hash Lemma

Application 1: Random Number Generation

Application 2: Quantum Key Distribution

Application 3: Learning With Errors

Further Works

References

Table of Contents

Introduction and Motivation

Bunch of Terminology

The Leftover Hash Lemma

Application 1: Random Number Generation

Application 2: Quantum Key Distribution

Application 3: Learning With Errors

Further Works

References

Introduction

- Let's begin with a simple puzzle :)

Introduction

- Let's begin with a simple puzzle :)
- Suppose I have a secret bit string of length 8, $b_1 b_2 b_3 \cdots b_8$.
- It doesn't serve real meaning, just for password purposes.
- You do *somehow* know that $b_1 \neq b_2$, $b_2 = b_3$, $b_4 = 1$.
- You want to find my bit string by brute-force.

Introduction

- Let's begin with a simple puzzle :)
- Suppose I have a secret bit string of length 8, $b_1 b_2 b_3 \cdots b_8$.
- It doesn't serve real meaning, just for password purposes.
- You do *somehow* know that $b_1 \neq b_2$, $b_2 = b_3$, $b_4 = 1$.
- You want to find my bit string by brute-force.
- How many possibilities do you need to check?

Introduction

- It’s not hard to see the answer is 32.

Introduction

- It's not hard to see the answer is 32.
- Here's an intuitive explanation.
- There are 8 unknown bits.
- There are 3 bits of information.
- Therefore, the answer is $2^{8-3} = 32$.

Introduction

- I feel kinda bad that you have some information on my secret.
- I think I should be able to "compress" my secret?
- Goal: You have no information about my secret.

Introduction

- I feel kinda bad that you have some information on my secret.
- I think I should be able to "compress" my secret?
- Goal: You have no information about my secret.
- Case 1: I know what information you have.
- Just change my secret to $b_1 b_5 b_6 b_7 b_8$.
- Now you know nothing, and there are 32 possibilities.

Introduction

- I feel kinda bad that you have some information on my secret.
- I think I should be able to "compress" my secret?
- Goal: You have no information about my secret.
- Case 1: I know what information you have.
- Just change my secret to $b_1 b_5 b_6 b_7 b_8$.
- Now you know nothing, and there are 32 possibilities.
- Case 2: I do not know what information you have.
- Now what?

Introduction

- I feel kinda bad that you have some information on my secret.
- I think I should be able to "compress" my secret?
- Goal: You have no information about my secret.
- Case 1: I know what information you have.
- Just change my secret to $b_1 b_5 b_6 b_7 b_8$.
- Now you know nothing, and there are 32 possibilities.
- Case 2: I do not know what information you have.
- Now what?
- This is where Leftover Hash Lemma comes in!

Motivation

- Q: Why do you feel the need to compress anyway?
- Q: Amount of brute-force remains the same.
- Q: Is rkm0959 delusional?

Motivation

- Q: Why do you feel the need to compress anyway?
- Q: Amount of brute-force remains the same.
- Q: Is rkm0959 delusional?
- A: partial leakage of secret information can be critical!
- A: brute-force is not the only possible attack in cryptography.
- A: here's a very good example in RSA.

Motivation

- Q: Why do you feel the need to compress anyway?
- Q: Amount of brute-force remains the same.
- Q: Is rkm0959 delusional?
- A: partial leakage of secret information can be critical!
- A: brute-force is not the only possible attack in cryptography.
- A: here's a very good example in RSA.

Textbook RSA

- Public Key: N and e with $\gcd(e, \phi(N)) = 1$.
- p, q are large primes, so factorization of $N = pq$ is hard.
- Private Key: d such that $ed \equiv 1 \pmod{\phi(N)}$.
- Encryption of m: $c = m^e \pmod{N}$.
- Decryption of c: $m = c^d \pmod{N}$.
- Factorization of N is equivalent to deriving d.
- Details: rkm0959.tistory.com/131

Motivation

Theorem (Partial Key Exposure : Coppersmith)

Let $N = pq$ be an n-bit RSA modulus. Then given the $n/4$ least significant bits of p or the $n/4$ most significant bits of p, one can efficiently factor N. Same applies for q.

Theorem (Partial Key Exposure : Boneh, Durfee, Frank)

Let $\langle N, d\rangle$ be a private RSA key in which N is n bits long. Given the $n/4$ least significant bits of d, one can reconstruct all of d in time linear in $e \log e$. Note that $e = 2^{16} + 1$ is usually used.

Motivation

- Q: Wait, why not just generate a new random secret?

Motivation

- Q: Wait, why not just generate a new random secret?
- A: First, generation of "truly random" bits is a hard task.
- A: Also, plz just wait until applications section :P

Motivation

- Q: Wait, why not just generate a new random secret?
- A: First, generation of "truly random" bits is a hard task.
- A: Also, plz just wait until applications section :P
- Also, it's not bad to be extra cautious in cryptography :)
- Cryptographers have very high standards in their schemes.
- Now that we have all the motivation, we begin.

Table of Contents

Introduction and Motivation

Bunch of Terminology

The Leftover Hash Lemma

Application 1: Random Number Generation

Application 2: Quantum Key Distribution

Application 3: Learning With Errors

Further Works

References

What Do We Want?

- Let's briefly summarize my goal, in simpler terms.

What Do We Want?

- Let’s briefly summarize my goal, in simpler terms.
- Then we will transform them into mathematical ones!

What Do We Want?

- Let's briefly summarize my goal, in simpler terms.
- Then we will transform them into mathematical ones!
- You have some information on my secret.

- I have no idea what information you have.

- At least I hope/know that you have no good guesses.

- I want to change my secret so that you have no info on it.

What Do We Want?

- Let's briefly summarize my goal, in simpler terms.
- Then we will transform them into mathematical ones!
- You have some information on my secret.
- **You have some probability distribution over my secret.**
- I have no idea what information you have.
- **I have no idea what your distribution is.**
- At least I hope/know that you have no good guesses.
- $\max_X P_{you}(\text{secret} = X)$ **should be small**.
- I want to change my secret so that you have no info on it.
- **I want your distribution to be close to uniform.**

Terminology Time

Definition (Min-Entropy)

Let X be a random variable. We define min-entropy $H_\infty(X)$ as

$$H_\infty(X) = -\log_2 \max_x P(X = x)$$

We say that X is a k-source if $H_\infty(X) \geq k$.

Terminology Time

Definition (Statistical Distance)

Let X and Y be two random variables with range U. The statistical distance between X and Y is defined as

$$\Delta(X, Y) = \frac{1}{2} \sum_{u \in U} |P(X = u) - P(Y = u)|$$

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

$$X \approx_{\epsilon} Y \iff \Delta(X, Y) \leq \epsilon$$

Terminology Time

- Nice! We got all the terminology needed :)
- Let's try to summarize what my goal is once again!

Terminology Time

- Nice! We got all the terminology needed :)
- Let's try to summarize what my goal is once again!
- You have a distribution over U that I don't know.
- X is a random variable that follows it.
- $H_\infty(X)$ is large enough for our argument to make sense.

Terminology Time

- Nice! We got all the terminology needed :)
- Let's try to summarize what my goal is once again!
- You have a distribution over U that I don't know.
- X is a random variable that follows it.
- $H_\infty(X)$ is large enough for our argument to make sense.
- I want to design a function $f : U \to \{0,1\}^n$
- f will be used to "compress" my secret.
- The resulting distribution is $f(X)$, and we want $f(X) \approx_\epsilon U_n$.
- U_n is a random variable, uniformly distributed over n bits.

Terminology Time

- Nice! We got all the terminology needed :)
- Let's try to summarize what my goal is once again!
- You have a distribution over U that I don't know.
- X is a random variable that follows it.
- $H_\infty(X)$ is large enough for our argument to make sense.
- I want to design a function $f : U \to \{0, 1\}^n$
- f will be used to "compress" my secret.
- The resulting distribution is $f(X)$, and we want $f(X) \approx_\epsilon U_n$.
- U_n is a random variable, uniformly distributed over n bits.
- I would also like n to be as large as possible.

Some Food For Thought

- What happens when $H_\infty(X)$ is small?

Some Food For Thought

- What happens when $H_\infty(X)$ is small?
- In this case you already have a great guess on my secret.
- I'm already doomed, so no need to do this anyway.
- You can also find some mathematical explanation.

Some Food For Thought

- What happens when $H_\infty(X)$ is small?
- In this case you already have a great guess on my secret.
- I'm already doomed, so no need to do this anyway.
- You can also find some mathematical explanation.
- Can we find a deterministic f that suits our needs?

Some Food For Thought

- What happens when $H_\infty(X)$ is small?
- In this case you already have a great guess on my secret.
- I'm already doomed, so no need to do this anyway.
- You can also find some mathematical explanation.
- Can we find a deterministic f that suits our needs?
- **No, and it can be proved. Left as exercise.**
- So we need f to be a random function!

Terminology Time

Definition (Randomness Extraction)

Let the seed U_d be uniformly distributed on $\{0,1\}^d$. We say that a function Ext : $\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ) extractor if for all random variables X on $\{0,1\}^n$ that are independent of U_d with the condition $H_\infty(X) \geq k$,

$$(\text{Ext}(X, U_d), U_d) \approx_\epsilon (U_m, U_d)$$

where U_m is uniform on $\{0,1\}^m$ and independent of X and U_d.

Done!

- Now we understand what my goal really is :)

Done!

- Now we understand what my goal really is :)
- Goal: Find a good extractor. How?
- Any educated guesses? Ideas?

Done!

- Now we understand what my goal really is :)
- Goal: Find a good extractor. How?
- Any educated guesses? Ideas?
- Idea: use **hashing**

Table of Contents

Introduction and Motivation

Bunch of Terminology

The Leftover Hash Lemma

Application 1: Random Number Generation

Application 2: Quantum Key Distribution

Application 3: Learning With Errors

Further Works

References

More Definition

Definition (2-Universal Hash)

A hash family $\mathcal{H} = \{h_i\}_{i=1}^{k}$, where $h_i : U \to \{0, \cdots n-1\}$ for each i, is *2-universal* if for all $x, y \in U$ such that $x \neq y$ we have

$$Pr_{h \sim \mathcal{H}}(h(x) = h(y)) \leq \frac{1}{n}$$

where $h \sim \mathcal{H}$ means that h is selected uniformly at random from $\mathcal{H}$.

Example

Consider $U = \{1, \cdots, N\}$ and $p \in [N, 2N]$. For all $a, b \in \mathbb{N}$, define

$$h_{a,b}(x) = (ax + b) \pmod{p} \pmod{n}$$

If we define

$$\mathcal{H} = \{h_{a,b} : 1 \leq a < p, 0 \leq b < p\}$$

we can show that $\mathcal{H}$ is a 2-universal hash family.

The Statement

Theorem (Leftover Hash Lemma)

Let X be a random variable with $H_\infty(X) \geq k$. Fix $\epsilon > 0$. Let $\mathcal{H}$ be a 2-universal hash family of size 2^d with output length $m = k - 2\log(1/\epsilon)$. If we define

$$Ext(x, h) = h(x)$$

then Ext is a $(k, \epsilon/2)$ extractor with output length m.

Proof Sketch

- View all distributions as a vector (probability mass function)
- Denote π_1 as the distribution of $(h(x), h)$
- Denote π_2 as the distribution of (U_m, U_d)
- Our goal is to bound $||\pi_1 - \pi_2||_1$.
- Part 1: Bound $||\pi_1||_2$ with the definition of 2-universal hash
- Part 2: Bound $||\pi_1 - \pi_2||_2$ with Part 1 and calculations
- Part 3: Bound $||\pi_1 - \pi_2||_1$ with Part 2 and Cauchy-Schwarz

Some Notes

- Why is this theorem so cool?
- Hashing in cryptography is a very tricky(?) subject
- One-Way-Functions, P=NP, etc... very hard
- 2-universal hash is relatively straightforward
- It's not hard to prove some hash family is 2-universal
- It's hard to prove that some hash is cryptographically secure
- We will see this come to use in Application 3.

Table of Contents

Introduction and Motivation

Bunch of Terminology

The Leftover Hash Lemma

Application 1: Random Number Generation

Application 2: Quantum Key Distribution

Application 3: Learning With Errors

Further Works

References

RNG is hard

Example (LCG)

The linear congruential generator is determined by the seed X_0, multiplier a, increment c, and the modulus m as

$$X_{n+1} \equiv aX_n + c \pmod{m}$$

However, it's easy to "predict" the RNG with a few known output values, making it inappropriate to use for cryptography.

Example (MT19937)

The frequently used RNG in Problem Solving, MT19937, can be also predicted with 624 consecutive output values. Therefore, it is also inappropriate to use for cryptography.

Randomness Extractors to RNG

- Where do we find a source of true randomness?

Randomness Extractors to RNG

- Where do we find a source of true randomness?
- Hardware: random noise (ex: thermal noise)

Randomness Extractors to RNG

- Where do we find a source of true randomness?
- Hardware: random noise (ex: thermal noise)
- But we don't even know the distribution of them?

Randomness Extractors to RNG

- Where do we find a source of true randomness?
- Hardware: random noise (ex: thermal noise)
- But we don't even know the distribution of them?
- We don't need to know it! Just use Leftover Hash Lemma.
- We only need the min-entropy to be large enough :)

Randomness Extractors to RNG

- Where do we find a source of true randomness?
- Hardware: random noise (ex: thermal noise)
- But we don't even know the distribution of them?
- We don't need to know it! Just use Leftover Hash Lemma.
- We only need the min-entropy to be large enough :)
- Weird Source of Randomness -> Leftover Hash Lemma (Randomness Extractor) -> Good Source of Randomness

Table of Contents

Introduction and Motivation

Bunch of Terminology

The Leftover Hash Lemma

Application 1: Random Number Generation

Application 2: Quantum Key Distribution

Application 3: Learning With Errors

Further Works

References

Basic Setups in QKD

- After main parts of the QKD procedure, we have the following
- Alice and Bob both have nearly identical shared keys
- Differences are caused by eavesdropping or simply errors
- We need to get rid of these errors to finalize the key!
- We also need to make sure Eve (eavesdropper) doesn't know non-negligible amount of information on the key.

Privacy Amplification

Information Reconciliation

Alice and Bob divides the key into blocks, and check for errors by sending parity bits. This is done over a **public channel**, so attackers can gain information by this process. After removing all errors, the two have the same key with high probability.

Privacy Amplification

Information Reconciliation

Alice and Bob divides the key into blocks, and check for errors by sending parity bits. This is done over a **public channel**, so attackers can gain information by this process. After removing all errors, the two have the same key with high probability.

Privacy Amplification

Alice and Bob now reduces Eve's partial information on the key. Eve already has some knowledge on the key via eavsdropping, and gained even more during the Information Reconciliation process. We may simply use Leftover Hash Lemma to do so.

Table of Contents

Introduction and Motivation

Bunch of Terminology

The Leftover Hash Lemma

Application 1: Random Number Generation

Application 2: Quantum Key Distribution

Application 3: Learning With Errors

Further Works

References

Introduction

We are given the following parameters.

- $n, m \in \mathbb{N}$, q, a prime number
- χ, a noise distribution over $\mathbb{Z}/q\mathbb{Z}$
- In practice, χ will be a discrete Gaussian distribution.

Definition (LWE Problem)

Let $s \in \mathbb{Z}_q^n$ be a "secret". We are given m samples of the form

$$(a_i, \langle a_i, s\rangle + e_i)$$

where a_i are selected randomly from $\mathbb{Z}_q^n$ and e_i is selected from χ. The search LWE problem $\mathrm{LWE}_{n,m,q,\chi}$ is to recover the secret s. The decisional LWE problem $\mathrm{LWE}_{n,m,q,\chi}$ is to decide whether a given set of samples are generated by the above method, or simply generated randomly from $\mathbb{Z}_q^{n+1}$. Note that without e_i, these problems can be easily solved with linear algebra.

LWE Facts

Theorem (Decision = Search : Informal)

If we can solve the decisional LWE problem with reasonable amount of samples efficiently, we can solve the search LWE problem with reasonable amount of samples efficiently as well.

LWE Facts

Theorem (Decision = Search : Informal)

If we can solve the decisional LWE problem with reasonable amount of samples efficiently, we can solve the search LWE problem with reasonable amount of samples efficiently as well.

Theorem (Average = Worst : Informal)

If we can solve the decisional LWE problem when the secret s is chosen uniformly random, we can solve the decisional LWE problem when the secret s is chosen from an arbitrary distribution.

LWE Facts

Theorem (Decision = Search : Informal)

If we can solve the decisional LWE problem with reasonable amount of samples efficiently, we can solve the search LWE problem with reasonable amount of samples efficiently as well.

Theorem (Average = Worst : Informal)

If we can solve the decisional LWE problem when the secret s is chosen uniformly random, we can solve the decisional LWE problem when the secret s is chosen from an arbitrary distribution.

Notes (Post-Quantum Crypto)

Unlike RSA, LWE and its variants (like Ring-LWE) have no known serious attacks by the use of quantum computing.

Public Key Encryption from LWE, Regev 2005

KeyGen

We are given $n.m, q, \chi$ where χ satisfies $||e|| \leq q/(4m)$ with high probability for $e \leftarrow \chi$. The KeyGen algorithm generates the secret key $sk = s \leftarrow \mathbb{Z}_q^n$ and samples $A \leftarrow \mathbb{Z}_q^{n \times m}$, $e \leftarrow \chi^m$ and sets the public key $pk = (A, s^T A + e^T)$. It returns the key pair (pk, sk).

Public Key Encryption from LWE, Regev 2005

Encryption

We have public key (A, b^T) and a message $\mu \in \{0, 1\}$. To encrypt, select a random $r \leftarrow \{0, 1\}^m$ and output the ciphertext

$$(Ar, b^T r + \mu \cdot \lfloor q/2 \rfloor)$$

Decryption

We have the ciphertext (u, v) and the secret key s. To decrypt, we note that the plaintext message can be retrieved as

$$\mu = \begin{cases} 0 & ||v - s^T u|| < q/4 \\ 1 & \text{otherwise} \end{cases}$$

Security Proof with Leftover Hash Lemma

- How do we prove the security of this scheme?
- Goal: after $poly(n)$ cases, we still cannot distinguish 0/1.

Goal

For any $k = poly(n)$, we must have

$$(pk, E(pk, \mu_1), \cdots, E(pk, \mu_k)) \approx (pk, E(pk, 0), \cdots, E(pk, 0))$$

where $\approx$ denotes "computational indistinguishability".

Security Proof with Leftover Hash Lemma

- In fact, it suffices to prove the following.
- Why? Think about triangle inequality :)

Simplified Goal

We must have

$$(pk, E(pk, 0)) \approx (pk, E(pk, 1))$$

where $\approx$ denotes "computational indistinguishability".

- In fact, such applications of triangle inequality have a name.
- It's called "Hybrid Argument", and we will use it now.

Security Proof with Leftover Hash Lemma

Hybrid 1 : Encryption of 0

$pk = (A, s^T A + e^T) = (A, b^T)$, $ct = (Ar, b^T r)$ where
$A \leftarrow \mathbb{Z}_q^{n\times m}$, $s \leftarrow \mathbb{Z}_q^n$, $e \leftarrow \chi^m$, and random $r \leftarrow \{0,1\}^m$

Hybrid 5 : Encryption of 1

$pk = (A, s^T A + e^T) = (A, b^T)$, $ct = (Ar, b^T r + \lfloor q/2 \rfloor)$ where
$A \leftarrow \mathbb{Z}_q^{n\times m}$, $s \leftarrow \mathbb{Z}_q^n$, $e \leftarrow \chi^m$, and random $r \leftarrow \{0,1\}^m$

Security Proof with Leftover Hash Lemma

Hybrid 1 : Encryption of 0

$pk = (A, s^T A + e^T) = (A, b^T)$, $ct = (Ar, b^T r)$ where
$A \leftarrow \mathbb{Z}_q^{n\times m}$, $s \leftarrow \mathbb{Z}_q^n$, $e \leftarrow \chi^m$, and random $r \leftarrow \{0,1\}^m$

Hybrid 2 : LWE Assumption

$pk = (A, b^T)$, $ct = (Ar, b^T r)$ where
$A \leftarrow \mathbb{Z}_q^{n\times m}$, $b \leftarrow \mathbb{Z}_q^m$, and random $r \leftarrow \{0,1\}^m$

Hybrid 4 : LWE Assumption

$pk = (A, b^T)$, $ct = (Ar, b^T r + \lfloor q/2 \rfloor)$ where
$A \leftarrow \mathbb{Z}_q^{n\times m}$, $b \leftarrow \mathbb{Z}_q^m$, and random $r \leftarrow \{0,1\}^m$

Hybrid 5 : Encryption of 1

$pk = (A, s^T A + e^T) = (A, b^T)$, $ct = (Ar, b^T r + \lfloor q/2 \rfloor)$ where
$A \leftarrow \mathbb{Z}_q^{n\times m}$, $s \leftarrow \mathbb{Z}_q^n$, $e \leftarrow \chi^m$, and random $r \leftarrow \{0,1\}^m$

Security Proof with Leftover Hash Lemma

- The punchline of this proof comes at Hybrid 3.

Hybrid 2 : LWE Assumption

$pk = (A, b^T)$, $ct = (Ar, b^T r)$ where
$A \leftarrow \mathbb{Z}_q^{n \times m}$, $b \leftarrow \mathbb{Z}_q^m$, and random $r \leftarrow \{0,1\}^m$

Security Proof with Leftover Hash Lemma

- The punchline of this proof comes at Hybrid 3.

Hybrid 2 : LWE Assumption

$pk = (A, b^T)$, $ct = (Ar, b^T r)$ where
$A \leftarrow \mathbb{Z}_q^{n\times m}$, $b \leftarrow \mathbb{Z}_q^m$, and random $r \leftarrow \{0,1\}^m$

Hybrid 3 : Pure Random

$pk = (A, b^T)$, $ct = (u, v)$ where
$A \leftarrow \mathbb{Z}_q^{n\times m}$, $b \leftarrow \mathbb{Z}_q^m$, and $u \leftarrow \mathbb{Z}_q^n$, $v \leftarrow \mathbb{Z}_q$

Security Proof with Leftover Hash Lemma

Theorem (Hash by Matrix Multiplication)

The distribution of (A, Ar) *is statistically* ϵ*-close to the distribution of* (A, u) *where* $A \leftarrow \mathbb{Z}_q^{n\times m}$, $r \leftarrow \{0,1\}^m$, $u \leftarrow \mathbb{Z}_q^n$, *as long as the inequality* $m > n\log(q) + 2\log(1/\epsilon)$ *holds true.*

Interpretation

We may prove this by interpreting the matrix A as a seed and matrix multiplication as hash. This can be proved to be universal.

Aftermath

- Conclusion: This is secure, but we need large m
- $m > (n+1)\log q + 2\log(1/\epsilon)$: quite large
- To make it more practical, we need smaller m
- Ideas: add more errors (LP11), remove LSBs (CKLS 16), etc...
- These encryption systems enjoy the homomorphic properties.
- The evolution towards FHE using LWE variants.

Table of Contents

Introduction and Motivation

Bunch of Terminology

The Leftover Hash Lemma

Application 1: Random Number Generation

Application 2: Quantum Key Distribution

Application 3: Learning With Errors

Further Works

References

Challenges in LHL

Large Entropy Loss

LHL begins with a distribution with min-entropy of k and ends with an output length of $k - 2\log(1/\epsilon)$. This means we suffer an entropy loss of $2\log(1/\epsilon)$. This is too large for many applications.

Large Seed Length

To apply LHL, we need to select a seed randomly. However, the seed length of a 2-universal hash function is not small. The seed length also grows with the number of extracted bits.

Recent Results : Large Entropy Loss

Theorem (LHL Revisited, 2011)

For a lot of applications, the entropy loss can be reduced to $\log(1/\epsilon)$*. With entropy loss L, the security with the LHL-derived key degrades from* ϵ *to* $\epsilon + \sqrt{\epsilon \cdot 2^{-L}}$*. Using this result, an efficient "general-purpose key derivation function" is built.*

Recent Results : Large Seed Length

Theorem (LHL Revisited, 2011)

The expand-then-extract approach is likely secure "in practice": one may generate the large seed with a small input seed and a pseudorandom generator. Also, this always works if the number of extracted bits is "small" i.e. logarithmic to the security parameter.

Table of Contents

Introduction and Motivation

Bunch of Terminology

The Leftover Hash Lemma

Application 1: Random Number Generation

Application 2: Quantum Key Distribution

Application 3: Learning With Errors

Further Works

References

References

- Regev, O. (2010). The learning with errors problem. Invited survey in CCC, 7, 30.
- Stinson, D. R. (2001). Universal hash families and the leftover hash lemma, and applications to cryptography and computing. Faculty of Mathematics, University of Waterloo.
- Barak, B., Dodis, Y., Krawczyk, H., Pereira, O., Pietrzak, K., Standaert, F. X., & Yu, Y. (2011, August). Leftover hash lemma, revisited. In Annual Cryptology Conference (pp. 1-20). Springer, Berlin, Heidelberg.
- Agrawal, S., Boneh, D., & Boyen, X. (2010, May). Efficient lattice (H) IBE in the standard model. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 553-572). Springer, Berlin, Heidelberg.
- a whole bunch of lecture notes