:0.5pt

KV@printgloss@

KV@printgloss@

format

format

format

format

An Undergraduate Guide to Lattice-based Cryptography with Applications in Fully Homomorphic Encryption

Submitted in fulfillment of English III *requirement*

for the

Bachelor of Science
in
Mathematics

Fall 2015

Submitted by

K. M. Short

Under the guidance of
Dr. Steve Butler

Department of Mathematics
Iowa State University of Science and Technology
Ames, Iowa – 50011

MSC (2010): 94A60, 14G50
Keywords: cryptography,lattices,convex bodies,
data encryption,coding theory,polytopes

Contents

1 Introduction

This paper contains a up-to-date, survey of lattice-based cryptography with an introduction to some information theoretic and security related topics. This survey begins at a high-level attempting to cover the minimal amount of terminology and background an undergraduate may need in order to start investigation into information theoretic and cryptographic fields; however, it is far from exhaustive, for a complete [1], field level and more exhaustive surveys I highly recommend [47] by Micciancio, and the most recent field level survey [55]. Both are quite dense, but more than worth the read.

This information is given in a format suitable for mathematics undergraduates who might wish to pursue such interests after graduation, or perhaps during an RUE. Movement proceeds towards results stemming from an incident in ideal ring-based fully homomorphic encryption called 'Soliloquy'. Soliloquy was a ring-based primitive constructed by the Government Communications Headquarters (GCHQ), which subsequently turned into a classical, and quantum attack on a particular subset of lattice-based schemes constructed over the much abused cyclotomic field of characteristic two.

The first sections give a brief introduction to classical cryptographic terminology and develops the concept of information theoretic security. The second section

FINISH!!!!

[1] but according to the author, Micciancio in some respects dated.

2 Notation

Before beginning please note the following notation as it will be used throughout the text.

The real coordinate space, or Euclidean space is denoted $\mathbb{R}^n$, and $\mathbb{Z}$ denotes the group of integers under addition. The usual variables denote real numbers x and y, if bold $\mathbf{x}$ denotes a column vectors, and $\curvearrowleft^t$, its transpose is a row vector. Complex variables are z and in certain specific contexts ω.

Matrices are denoted with uppercase letters A or B.

The variables p and q are positive integers. Let q be a positive integer then group of integers modulo q, is denoted $\mathbb{Z}_q$.

3 Another Note

It may be worth noting that although cryptography can be approached from a pure mathematical perspective, (indeed the definitions for security themselves stated in terms most mathematics undergraduates would find familiar and comfortable) it is still an applied topic, where certain rules, norms should be expected when attempting to move into this small, but closely knit community of applied and theoretical cryptographers.

It is also worth noticing that the divide between civilian and agency cryptographers has of late widened. Previously, the cryptographers employed by governmental agencies both in the US and several other countries such as the UK, and New Zealand were more or less concerned with the same outcomes. Attitudes for the most part remained cooperative due to an mutual goal of reducing threats and increasing the security of all systems.

However, recent developments have lead to feelings mistrust and irritation towards these agencies. In particular, the revelation that some governmental agencies may have intentionally subverted cryptographic standards e.g., encouraging the NIST to adopt and promote a 'random' number generator for dual elliptic curve cryptography, which turned out not only far from random but known to be significantly flawed many times over[38] [37] [71].

It is enough regardless of ones stance on mass surveillance and related issues to feel a certain amount of mistrust towards a party who suddenly adds a non-arbitrary amount of added work to an already overflowing stack. his is similar in a sense, to cleaning up ones yard at the same time as your neighbor and upon returning from the backyard believing you have finished, finding the same neighbor has simply blown all their leaves into your yard. Uncomfortable as it may be to deal with political untidiness of this sort it is better that you seek out information and decide for yourself which side of a table you are comfortable sitting than to be surprised later. This being said you are once again reminded that cryptography is concerned with the transmission, verification, and overall security of arbitrary date. Be it your grandmothers cookie recipe or a bank statement the outcome of a certain scheme should be the same. The end goal is an increase in security not an increase to insecurity, political intrigue, or any other uncommon quantity; put the spy movies down[2].

4 Introductory Cryptographic Terminology and Notation

Before we begin constructing a scheme we need to know which type of channel will be used to transmit the communications. Not all schemes will be appropriate for a particular channel. Choosing appropriately for the users and channel instead attempting to force a scheme to fit an unsuited one will not only save a lot of wasted effort during implementation it also ensures that an appropriate basis for security is in place prior to the implementation.

4.1 Basic Communications Channels

Definition 1. *Confidential Channel A confidential channel enables communication that is resistant to interception.*

Definition 2. *Authentic Channel An authentic channel allows communication which resists alterations by unauthorized users.*

Definition 3. *Secure Channel A secure channel enables communication which may resist both interception, and alteration by unauthorized users.*

There are many more different types of channels which are of cryptographic and information theoretic interest. The above channels provide a pretty good starting point to serve as our frame of reference.

4.2 Asymmetric and Symmetric Key Schemes

Modern cryptographic schemes can generally be divided into two classes: asymmetric key encryption schemes and symmetric-key encryption schemes.

Symmetric-key encryption schemes involve the use of only one key, used by all communicating parties called a *shared secret.* In contrast, asymmetric key encryption schemes make use of one key pair for each user, composed of a secret key and a public key; in general they tend to be more efficient than asymmetric schemes but have the disadvantage of requiring users have an apriori knowledge

[2]Avoid using the word 'cyber' at all costs.

of the need to communicate. Specifically, symmetric-key schemes require that secret keys be distributed over a secure channel[3].

Symmetric key distribution schemes (when used alone) are not typically appropriate for networked environments, which depending on context may constitute an insecure channel.

Remark. It is worth explicitly stating that any type of cryptosystem regardless of the number of keys or method of generating keys becomes insecure (in whole or in part) when a secret key is compromised. Additionally it is an open problem, that in any given channel (regardless of resistance type), there is no known long-term solution which allows a user to know if a channel communicated, or failed to communicate securely. [4]

In contrast, public-key cryptography is an asymmetric-key class of cryptosystems which does not require any foreknowledge of the need to communicate between users, nor does it require a secure channel. PKC solves the problem of how to enable secure communication between two people who have never met over insecure channels.

A public-key scheme makes use of one key pair per user, consisting of a private (secret) key and a public key. Users wishing to communicate make use of public keys to encrypt messages; and private keys to decrypt messages.

In general PKC depends on the following assumptions:

Assumption 1. In the forward direction, we assume there exists a one-way function which is 'easy' to compute.

Assumption 2. In the reverse direction, we assume that the same function is 'hard' to invert.

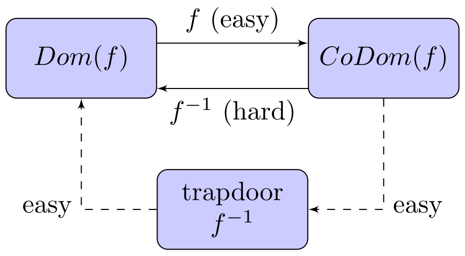


Figure 1: Oneway function and trapdoor

[3]In real world applications there is no such thing as a secure channel.

[4]The given definitions for channels are extended under the subfield of *Universal Decomposability*; typically the definitions are given the prefix 'ideal', followed by the channel name in order to distinguish from these more general forms.

4.3 Complexity and Cryptography

Informally, we say that a function is 'easy' to compute if an algorithm can provide a solution in a relatively short amount of time [5]. Similarly, we refer to a function as being 'hard' to compute if it cannot provide a solution in a more than reasonable amount of time. The ease or difficulty in finding solutions to a given function is the main focus of computational complexity.

In cryptography the hardness assumption involved in finding a solution for a given function quantifies the chance of finding a solution to an assumed or known hard problem. In cryptography hardness quantifies the chance an adversary has of finding a solution to the mathematical problem a cryptosystem is based upon; finding a solution may give rise to the recovery of a secret key, or decryption of a message (without recovering a key).

The common assumptions attributed to an adversary are as follows:
Assumption 3. The adversary is in possession of (or has access to) slightly more than a reasonable (but still finite and non-quantum) amount of computational resources;
Assumption 4. If the adversarial system (as described above) provides a solution to the problem serving as a basis for security of the target system, it does so by applying a probabilistic, polynomial-time, algorithm.

4.4 Formal Definitions of Asymmetric & Symmetric Cryptography

The concepts of symmetric and asymmetric cryptographic-key schemes are given below independently. A typical cryptosystem mixes more than one type of cryptographic primitive, depending on the functions required.

Both asymmetric and symmetric key schemes discussed here are assumed to be classical cryptosystems. By 'classical' we mean the systems are not quantum computers (or algorithms), although they may be parallel clusters.

4.4.1 Symmetric-key Cryptography

In canonical symmetric-key schemes only one key is generated for both the encryption and decryption algorithms. Symmetric-keys must therefore be exchanged over secure channels where both communicating parties have agreed in advance to the method of encryption and have decided upon a shared secret key.

[5]Reasonably short meaning well before the heat death of the universe. More than reasonable resources suggests allowing amounts upto and possibly including the heat death of the universe; possibly several times.

Formally, we say the two parties are in possession of a *shared secret* and each party is mutually responsible for the maintenance of this secret.

Drawbacks of symmetric key schemes include the use of a secure channel, and a single shared secret key.

We define a symmetric-key scheme as follows:

Definition 4. *Keyspace Let $\mathcal{K}$ be the set of all possible keys k generated by a key generation function* **Gen***, called the* keyspace*.*

Definition 5. *Key Generation Function The key generation algorithm* **Gen** *is a probabilistic function which uses an appropriately chosen uniform distribution to generate the elements of the keyspace $\mathcal{K}$.*

The key generation algorithm **Gen**, has uniform distribution from which key elements k must be chosen randomly.

Definition 6. *Message Space The message space, denoted $\mathcal{M}$, is the set of all possible messages m, also called the* plaintext space*.*

Definition 7. *Encryption Function The encryption function* **Enc** *takes a message m as input, and uses the secret key k to produce a ciphertext c.*

Definition 8. *Ciphertext Space The set of all possible ciphertexts (or the ciphertext space) is denoted $\mathcal{C}$, is defined by the pair of spaces $(\mathcal{K}, \mathcal{M})$.*

Definition 9. *Decryption Function The decryption function* **Dec***, takes a ciphertext c as input and uses the secret key k to recover the message m.*

Symmetric Key Cryptosystem

Taking all components together, we have a typical *symmetric-key* cryptosystem. We use cryptography's favorite archetypes Alice and Bob, and their eavesdropper Eve to enact the scheme.

A symmetric-key cryptosystem is given by a set of algorithms $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ and the message space $\mathcal{M}$.

Alice and Bob make use of the cryptosystem as follows:

First, Alice and Bob use **Gen** to produce a shared secret key k.

Whenever Bob wishes to send a message m to Alice (or Alice to Bob) he inputs the pair (m, k) into **Enc** and receives a ciphertext of his message c.

$$c := Enc_k(m)$$

When Alice receives the ciphertext c from Bob she inputs the pair (c, k) into the decryption function **Dec** and retrieves the plaintext message m.

$$Dec_k(c) := m.$$

To ensure the correctness of a cryptosystem, we show there exists a bijection between the one-way function and the trapdoor function.

Meaning composition of the functions **Enc** and **Dec** will return the same outputs for a given input, for every message key pair (k, m) : $k \in \mathcal{K}$ and $m \in \mathcal{M}$.

That is, the relation **Enc** : $\mathcal{M} \times \mathcal{K} \to \mathcal{C}$ is a mapping of $m \mapsto c$ for every $c \in \mathcal{C}$, if and only if **Dec** : $\mathcal{K} \times \mathcal{C} \to \mathcal{M}$ is also a relation mapping such that $c \mapsto m$ for every $m \in \mathcal{M}$.

Thus we have shown for our toy[6] symmetric-key cryptosystem the relations **Enc** and **Dec** are closed under composition, and return the expected outputs for all m.

$$Dec_k(Enc_k(m)) = m$$

The composition of the **Enc** and **Dec** functions produces in one direction the original message, and in the other direction we get the ciphertext,

$$Enc_k(Dec_k(m)) = c.$$

Therefore we have confirmed the **Enc** function has an inverse (is reversible), likewise for the **Dec** function.

[6]And general too!

4.4.2 Asymmetric Key Cryptography

In contrast, asymmetric-key schemes are determined by two sets of keys one for each party. Each set consists of a *private key*, s_k known only to the owner and a *public key*, p_k which can be widely distributed.

Asymmetric-key cryptography form the basis for all Public Key Cryptosystems (PKC) such as RSA, and the Diffie-Hellman protocol.

A public-key uses a One-way function as its public key, p_k; this function is easy to compute but hard to invert and is available publicly for use as an encryption method for the key owner.

A second key called the private key, s_k is known only to the key owner; this key is a 'Trapdoor Function', which provides some additional information allowing the inverse of the one-way function to be more easily computed.

Note 1. The One-way Function Assumption is an open question in computability and complexity theory. It is common to make the assumption that one-way functions exist, in order to create new cryptosystems.

Definition 10. *Key Element Let s_k be a private key corresponding to a p_k. A key element of a public-key scheme is then defined as follows:*

$$k = (s_k, p_k).$$

The definitions for the keyspace $\mathcal{K}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$ are then defined in the same way as symmetric-key schemes. However, the definition of the cryptosystem is slightly different.

Definition 11. *Asymetric-key Cryptosystem An asymmetric-key cryptosystem is given by a set of algorithms* $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ *along with the message space $\mathcal{M}$, keyspace, $\mathcal{K}$, and ciphertext space $\mathcal{C}$*[7]*.*

Let $k = (s_k, p_k)$ be a key element such that s_k is an element of the set of all possible keys $\mathcal{K}$, such that s_k is chosen randomly from a

[7]It would seem at first glance that asymmetric and symmetric cryptosystems are isolated methods of encryption. However, symmetric encryption is well-suited for the task of assisting asymmetric key schemes not only for efficiency purposes but also in the role of key management.

space which generated in a uniform manner from an appropriate distribution.

$$\forall p_k \in \mathcal{K} \, \exists \, \mathbf{Enc}_{p_k}(\mathcal{M},\mathcal{C}) : \mathcal{M} \times \mathcal{K} \to \mathcal{C}.$$

$$\forall p_s \in \mathcal{K} \, \exists \, \mathbf{Dec}_{p_s}(\mathcal{C},\mathcal{M}) : \mathcal{K} \times \mathcal{C} \to \mathcal{M}.$$

If $k \in \mathcal{K}$*, then the composition* $\mathbf{Dec}_{p_k}(\mathbf{Enc}_{p_s}) := m, \forall m \in \mathcal{M}$.

5 Semantic Security: Defining Security with Games

Semantic Security was a game proposed by Goldwasser and Micali in their 1982 paper entitled: *'Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information'*[35], in response to papers from Rivest, Shamir, Adelman, [64] and Rabin [60] detailing the theory behind the Diffie-Hellman based RSA cryptosystem.

Goldwasser and Micali's intention was to strengthen the security assumptions of Diffie-Hellman based cryptosystems by formally defining what it would mean for such systems to be called secure.

5.1 Goldwasser & Micali's: Mental Poker and Partial Information

[[35]] Proposed the following property for any implementation of a Diffie-Hellman public-key cryptosystem:

Proposition 1. *'An adversary, who knows the encryption of an algorithm and is given the ciphertext, cannot obtain any information about the cleartext.'*

[35]

Informally, the above property states that a given cryptographic scheme is considered insecure, if given some of the ciphertext (but not the private key), it is possible for an adversary to recover *any* information about the plaintext, or if the adversary can recover useful information about the plaintext of the message, by some manipulation of the ciphertext in a reasonable amount of time (feasibly).

5.2 Weaknesses in the Assumptions of RSA

Goldwasser and Micali pointed out that the security assumptions given in [60] and [64] had some particularly significant drawbacks. The orignal construction of RSA was unpadded, only relied on a random choices for messages.

Assumption 5. There exists a Trapdoor Function $f(x)$ that is easily computed, while x is not easily computed from $f(x)$ (unless some additional information is known). To encrypt any message m, you evaluate f at m, and receive the ciphertext $f(m)$.

This first assumption gives us the following objection and cases.

smallskip
Objection 6. If the inverse of f has some specialized form, or some additional structure, simply assuming that $f(x)$ is assumed to be a trapdoor function does not guarantee that x cannot be easily computed when this additional information is not known.

Case 1. First, messages do not typically consists of randomly chosen numbers, they have more structure.

Case 2. Second, the additional structure these messages possess may assist in decoding the ciphertext.

Example 1. Compare two functions $f(m)$ and $g(n)$, where the input m for f is some random ASCII sequence, and the input n for g is an ASCII sequence which represents sentences written in English. Then it may happen that m is hard to recover from $f(m)$, but that n is easy to recover from $g(n)$.

Remark. Even if we assume that f is a trapdoor function, that does not negate the possibility that the message or even half the message may not be recoverable from $f(x)$.

Concerning the use of trapdoors in general:

> ‘Covering ones face with a handkerchief certainly helps to hide personal identity.
>
> However: it will not hide from me the identity of a special subset of people: my mother, my sister, close friends.
>
> I can gather a lot of information about people I cannot identify: their height, their hair color, and so on.’

These facts about the use of a handkerchief are exactly those you would find issue with in using any public-key cryptosystem whose security relied soley on trapdoors. In fact, they are the same arguments which arise as problems in cases one and two.

In order to better understand how this analogy can provide a better definition for the security of a cryptosystem, Goldwasser and Micali switch RSA from using only a randomly choosen message the set of all possible messages in the message space.

Claim 1. If the set of messages $\mathcal{M}$ is sparse in $\mathbb{Z}_N^*$, then the ability to correctly recover even 1% of all messages is not possible for a random polynomial time algorithm in the case of factoring.

Sparse in this context means that when $x \in \mathbb{Z}_N^*$ is chosen randomly, we have a probability of any x chosen being a message is for all intents and purposes zero.

The requirement that the messagespace should be sparse means that encoding a message does not simply rely on the content of the message but also the outcome of a sequence of fair coin tosses. The addition of the coin toss outcomes expands the size of the number of possible encodings of a message, where each possible encoding remains unique.

Second, they define the notion of secure for any public-key cryptosystem in a way that disallows the use of trapdoor functions:

Definition 12. *Let $\mathcal{P} \subset \mathcal{M}$ be an arbitrary decision problem which is easy to calculate, and let $m \in \mathcal{M}$.*

If an adversary has the encrypted form of m and is able to calculate $\mathcal{P}(m)$, then the adversary can obtain some *information about m from its ciphertext.*

The first given definition is necessary but it is not sufficient.

RSA defined purely in terms of trapdoors is deterministic. Meaning, if an adversary has some amount of ciphertext and can calculate $\mathcal{P}$, then they might be able to just guess the message m.

However, if $\mathcal{M}$ is an appropriate uniform distribution then calculating $\mathcal{P}(m)$ if now phrased in terms of a probability. This addition to the randomization, strengthens the notion of semantic security and gives us the following stronger notion of security.

Definition 13. *Semantic Security Let ε be the advantage ascribed to an adversary in solving $\mathcal{P}$. If the adversary has a probability greater than $p+\varepsilon$ of solving $\mathcal{P}$ then we should consider the system insecure.*

In other words, a cryptosystem is secure if an adversary cannot *feasibly* gain any information about the plaintext by manipulating the ciphertext.

Definition 14. *Perfect Secrecy (Shannon) If an adversary in possesion of some ciphertext cannot gain any additional information about the plaintext (message), then we call the cryptosystem* perfectly secure*.*

In [35] the definition of semantic security is a slightly weaker form of Shannon's *perfect secrecy.* The key difference is that semantic security relaxes the restriction by Shannon from if an adversary cannot gain *any* information about the message, to the adversary cannot gain any information about the message *feasibly.*

$$\text{semantic security} \xrightarrow{\mathcal{A}(c)} \mathcal{M}$$

$$\text{perfect secrecy} \overset{\mathcal{A}(c)}{\not\longrightarrow} \mathcal{M}$$

Another noticeable feature of semantic security is that it only can be applied to *passive adversaries*, i.e. adversaries which may generate and study ciphertexts from some public key. It does not apply to the case where the adversary may request a decryption of chosen ciphertexts[8].

Given these (and a few unrelated) reasons it becomes evident that semantic security is an insufficient condition for ensuring the security of a cryptosystem. However, semantic security provides a bound on the minimum amount of security a cryptosystem should be able to provide.

5.3 'Provable' Security

It is not difficult to understand why an object may need to be formally defined before more work can proceed, particularly in the case of more abstract concepts such as security.

[8]It also does not apply to cases in which someone encrypts their own key, i.e., circular security.

Example 2. Take for example the concept of safety in the context of bridge construction. It is easy to spot the case where the bridge is definitely not safe.

That's the case where a person who entered the bridge did not arrive at the otherside (or perhaps at all) by continuing to use the bridge, i.e., the bridge failed.

It is much harder to consider all the cases where a trip over the bridge might range between perfectly safe to not at all.

There might be many cases where the person arrived at the other side of the bridge but not entirely safely.

In some cases it will be the fault of the bridge, in other cases it may be other travelers, and in some cases it might be the person themself.

However, it is clear for any case other than bridge failure we may claim the trip was safe. Clearly we need contextually related definitions for the notion of safe in the case of crossing bridges.

In the same way, it is necessary for there to be contextually relevant notions of security, it is also necessary that these notions be quantifiable in someway in order to build safe guards against insecurities.

In cryptography we use the method of *security reduction*, or proof to determine the level of security a given cryptosystem imparts. This reduction is accomplished in many way but in general the process is as follows:

- Remove all cases where the system is found insecure due to side channel or implementation fault based attacks[9].
- Define the capabilities of the adversary (adversarial model).
- Define what level of access the adversary is assumed to have.
- Define the level of computational resources you may assume the adversary has at their disposal.

[9]Usually these cases either cannot be accurately modeled or are specific to the implementation.

- State the computational hardness of the mathematical problem your scheme will base its security upon (why is it an appropriate choice?)
- If you can then state the security requirements of your scheme formally and prove the hardness of the mathematical problem is by the given assumptions on the adversary infeasible, you have 'proved' your scheme.
- Alternatively, if you can find a scheme similar to your own whose hardness has already been proven, you may be able to reduce the proof of your systems hardness on some case where the other has been shown to hold.
- It might be then possible to state that assuming the hardness of the other problem your problem is at most as hard (or at least) as the other problem.

5.3.1 Mathematical Proofs vs. Security Proofs

It is important to understand that a mathematical proof and a security proof are not the same kinds of animals. While a mathematical proof gives an absolute assurance of some objects existence, properties, or some other notion. A 'proof' of security does not offer come with any such resolve due to its reliance of the hardness or intractibility of a given mathematical problem.

One should not make the mistake in thinking a certain cryptographic scheme is equivalent to the intractability its corresponding mathematical problem.

The relationship between the security of a system and it's corresponding underlying problem is not at all bidirectional.

Specifically, if $\mathcal{P}$ is some intractable problem serving as a basis for a cryptosystem $\mathcal{S}$. The intractable property of $\mathcal{P}$ is not a sufficient condition to ensure the security of $\mathcal{S}$.

Example 3. The RSA algorithm's method of encryption uses modular exponentiation by some fixed prime, this function has a corresponding inverse which decomposes the modulus into its prime factors. The decryption algorithm of RSA takes advantage of the existence of its inverse. However, it has never been proven that factoring the modulus is the only way or even the best way to decrypt the messages encoded by the modular exponentiation function. We assume that solving the underlying problem of RSA is equivalent to factorization, but this may not be true at all. In fact there have been results on both sides of the argument in many papers. If we consider [**?**], then we have a result which implies that the fixed prime which serves as the exponent for RSA may be a much easier way of breaking RSA than attempting to decompose it into prime factors. If we consider [26], we might have an implication that RSA is equivalent to factorization.

However, the cases we choose to compare may have no relation on at all to the situation that really might exist.

It would seem that it might be worth investigating if the questions of equivalency to factorization are the right type of questions to ask in this case, at all.

6 Attack Models

6.1 Known Plaintext Attack (KPA)

The adversary in this case has at least one pair of corresponding plaintext and ciphertext that they did not choose. The adversary knows the context or meaning of the plaintext. The adversary does not have access to the system and cannot generate more pairs of plain and ciphertext; they must work only with what they have.
Assumption 7. The adversary knows the system.
Assumption 8. The adversary has one ciphertext and a corresponding ciphertext.

Goal: The adversarys job is then to determine the decryption key.
Example 4. Eve has heard a conversation between Alice and Bob and knows that they will meet each other 'next Monday at 2pm'. This particular conversation that Eve overheard is encrypted and Eve has the ciphertext and she knows what was said so she has the plaintext. So Eve knows somewhere in the ciphertext is some string corresponding to 'next Monday at 2pm'. Eve can then use that she knows this string to decode some portion of the ciphertext and from there try to deduce the secret key.

Specific Examples

Example 5.
- Cesar Ciphers
- The Polish Break of the Enigma Machine
- The Bletchley Park Break of the Enigma Machine
- Old versions of PKZIP encrypted with AES

6.2 Chosen Plaintext Attacks (CPA)

Chosen Plaintext Attack I (Batch Chosen Plaintext Attack)

Assumption 9. The adversary knows the system.
Assumption 10. The adversary has temporary access to the encryption mechanism containing the key.

Goal:

For the time the adversary has access to the mechanism they must choose plaintexts, construct the corresponding ciphertexts and deduce the decryption key somewhere in this or after this process.

Chosen Plaintext Attack II (Adaptive Chosen Plaintext Attack)

Assumption 11. The adversary knows the system.
Assumption 12. The adversary has unlimited access to the encryption mechanism containing the key.

Goal:

The adversary can then choose plaintexts, see what the system returns, intelligently decide upon and submit more plaintexts to the system and use the constructed corresponding ciphertexts to deduce the secret key (trial and error).
Example 6. A very badly designed email service for some company provides encryption of messages, however it only generates one key per branch.

Eve knows that Alice and Bob work at a particular branch of a company gets a job at the company transfers to that branch.

Eve uses the same email service sends email with content she chooses through the service (perhaps to herself BCCing or CCing herself it doesnt matter, Eve chooses the content), Eve then analyzes the ciphertexts of the emails she sent deduces the secret key and then proceeds to decrypt messages belonging to Alice and Bob.

6.3 Chosen Ciphertext Attacks (CCA)

Chosen Ciphertext Attack I (Lunchtime Attack)

Assumption 13. The adversary knows the system.
Assumption 14. The adversary has temporary access to the decryption mechanism containing the key.
Assumption 15. The adversary has one or more ciphertexts which correspond to this key.

Goal: The job of the adversary is generate the plaintext corresponding to the ciphertexts (easily done), and deduce the key.
Example 7. Eve has acquired some encrypted messages sent from Bob to Alice. She has gained access to Bobs computer (the decryption oracle) while Bob is at lunch. Eve gets access to Bobs computer while he is at lunch and decrypts the set of encrypted messages. Eve then uses the ciphertexts and plaintexts to deduce the secret key.

Chosen Ciphertext Attack II (Adaptive Chosen Ciphertexts Attack)

Assumption 16. The adversary knows the system.
Assumption 17. The adversary has unlimited access to the decryption mechanism containing the key.
Assumption 18. The adversary has one or more ciphertexts which correspond to this key, and can now generate more whenever they choose.

Goal: The job of the adversary is to generate the plaintext corresponding to the ciphertexts (easily done), use the information to intelligently choose more plaintexts to generate ciphertexts and in this process deduce the key.

7 Complexity and Cryptography

In computational complexity functions can be viewed as sets of questions, the solutions of a function can be seen as an answer to the question posed by a function.

Problems (Questions) can used as input for algorithms which may confirm, deny, or even lie concerning the answers of a given question. We also may ask questions about the workings of the algorithm itself. We might be concerned with how the algorithm attempts to find a solution of a given problem. We can ask how efficiently the algorithm may (or may not) find a solution for a problem; we may want to know the amount of resources an algorithm requires as it attempts to find a solution.

We of course would like to know if even when given these resources if the algorithm will be able to find any solution to the problem at all.

7.1 Computational Problems

The set of all algorithms can be divided into two general types: deterministic, and nondeterministic.

Definition 15. *Deterministic Algorithm A deterministic algorithm is one which always produces the same output for a particular input. More formally, a deterministic algorithm is a mathematical function which takes a particular but arbitrarily chosen element from it's domain (the set of all inputs for a function) and maps the element uniquely to an element in the functions codomain (the set of all outputs).*

In other words, a deterministic algorithm is one which always traverses the same path in computing an output for a given input, the result is that a particular input will always land on the same output.

In contrast, an algorithm which is nondeterministic provides no guarantee that a particular input element will produce a unique output element.

Definition 16. *Nondeterministic Algorithm A nondeterministic algorithm is a probabilistic algorithm which may produce different outputs each time it runs due to the fact the algorithm is permitted to make different choices during execution. The output the algorithm returns may be unique or it may not, a result of the dependence of the algorithm on a random number generator.*

A nondeterministic algorithm may or may not travel the same path while it computes an output, it may have many more routes available to choose from than a deterministic algorithm.

Deterministic and nondeterministic algorithms have different uses stemming from their differences in output. A deterministic algorithm might be an ideal choice when the problem you want to solve is a well-defined mathematical function, say $f(x) = y$. However, if the problem may have multiple solutions each one ranging in 'correctness', a nondeterministic algorithm may provide a more appropriate mechanism for finding these solutions.

An example of a problem which might be aided by a nondeterministic algorithm is a path selection problem, where the goal is to reach a particular destination, but along the way you might have to choose which direction you wish to go when you reach a crossroads.

A computational problem is one where finding a solution may be aided using a computer, that is one in which it would be impractical to just grab a scrap of paper and scratch out a solution. These types of problems are classified by the level of difficulty in finding a solution.

The difficulty may be quantified in terms of resources, time, communication, or components an algorithm may use while it runs. This is called the *complexity class* of the algorithm.

In application, an instance of a problem may be given in terms of bit strings where the alphabet is binary, (i.e. the set $\{0, 1\}$), hexadecimal, or some adjacency matrix.

When phrasing a theoretical problem for use in a computational problem, it is typical to define the concept generally; this enables the problem instances to be represented in a way which ensures it can be transformed between different choices of encodings in applications.

Computational problems typically fall into three basic representations: decisional, search, and promise, described below.

Decision problems are instances where a question can be answered positively or negatively.

For example, we might want to know if an element is a member of a set. If the element is in the set the algorithm returns yes, else it returns no.

Definition 17. *Decision Problem (informal) In a formal system, a decision problem is a question which can be answered as* yes *or* no, *depending on the input.*

Definition 18. *Decision Problem Typically the set of outputs for an algorithm providing a set of solutions for a decision problem is defined to be the subset of all solutions for which the answer was positive, or for which the returned value is* 'yes'.

Decision problems in computational complexity can be further classified based on the level of difficulty (given in terms of resources, or some other value), and assuming the most efficient algorithm is used to find a solution for the problem instance.

In most cases it is possible to describe a solution for a decision problem, but impossible to construct an algorithm which can produce such a solution, these problems are called *undecidable.*

Definition 19. *Decidable Problem A decision problem $\mathcal{P}_d$ is decidable if it is a* recursive *or computable set. In terms of algorithms, decidable refers to a problem which can be solved in a finite amount of time, correctly.*
Definition 20. *Undecidable Problem A decision problem $\mathcal{P}_d$ is called undecidable if it's statement is neither provable nor refutable.*

It is often useful to know if a particular algorithm can be constructed or give a mathematical model of a particular algorithm, simulations such as these are usually defined over Turing machines. Informally, a Turing machine is a mathematical model which is capable of simulating an algorithm.

The following theorem is helpful in describing the remaining representative problems, as well as subsequent content within this paper.
Theorem 1. *Church's Thesis (Strong Version) Any physical computational device can be simulated by a Turing machine in polynomial time.*
Summary 1. Any computational device uses an amount of memory which grows linearly with respect to the computation.

In recursion theory, the case where a problem is classified as undecidable is measured by it's *Turing degree.*
Definition 21. *Turing Degree The Turing degree of an undecidable problem measures the noncomputability in determining a solution.*

Example 8. Decision problems are also used for optimization, or finding the *best answer* for a given problem. There are many standardized methods describing transformation between optimization problems and decision problems; many can be constructed without a significant increase to the Turing degree.

A search problem is an algorithm which verifies the existence of some object. These algorithms are typically stated as Turing machines (informally a model which is capable of simulating an algorithm).
Definition 22. *Search Problem Let T be a turing machine, and $\mathcal{R}$ a binary relation between $x \sim y$. T solve $\mathcal{R}$ if:*

If given an x there exists a y such that $\mathcal{R}(x, y)$, then T accepts x as a member of $\mathcal{R}(x, y)$ and produces an element z such that $\mathcal{R}(x, z)$.

If T cannot find a y when given an x which satisfies the binary relation, T rejects x and halts

In other words, a search problem is one where an algorithm is not used, and a problem is not solved, but where a description of a solution to a problem is shown to exist, or not.

A search problem can provide a description of a solution to a given decision problem. In fact, for every search problem there is a decision problem which corresponds to it.

For example, a search problem could find that at least one element in a subset satisfies a particular binary relation, but since it only needs to find out if a particular kind of element exists it may stop after finding only one.

In contrast a decision problem compares all elements in the subset with the description given by the search function and returns a subset of all the elements which matched the description.

A search problem may be defined as a starting or end state, a transformation between states, or as a successor function.

A promise problem is a general form of a decision problem which produces a more ambiguous output.

Where a decision problem does not exhaust the set of all possible solutions by answering each input in either the negative or affirmative a promise problem acts on the set of all possible solutions. The output of a promise algorithm for a particular input can tell you which subset a solution lies in, it may lie, or it may not answer at all.

In particular, if the solution for a given input is a member of the positive or negative subsets the promise problem outputs which subset it is a member of. However, if the input corresponds to a member which does not lie in either the positive or negative sets the promise problem may output anything at all or it may never produce an output (does not halt).

7.2 Complexity Issues

For nearly every classical cryptographic construction of the object is as follows:

First, generate a specific instance of a problem and it's solution in the complexity class NP, such that it is reasonable to assume the problem is hard to solve; where hard to solve is meant in the worst-case.

Take again as an example, the integer factorization problem and the RSA cryptosystem.

Let $k \in \mathcal{K}$ be a public key generated by a probabilistic key generation function **Gen**, where k is chosen randomly from a uniform distribution of the $\mathcal{K}$ (the keyspace).

$$(m, k) \leftarrow \mathbf{Gen}$$

Let m be an arbitrarily chosen element of the message-space $\mathcal{M}$, and let c be the corresponding ciphertext of $\mathcal{C}$ for m.

Choose $p, q \in \mathbb{P}$ (the set of primes), in a random way.

Construct the product $N = pq$, and define it to be the modulus.

If we have chosen p and q nicely, then we hope we will have a high-confidence which provides some assurance that p and q will not be ε close (approximately equal).

That is, the product $pq = N$ does not give us p, q both with $\varepsilon > 0$ close of $\sqrt{N}$, as this would make N easily factorable into p and q.

Next, choose e such that

$$e \perp \phi n = \phi(p)\phi(q) = (p-1)(q-1)$$

, i.e. e relatively prime to ϕn.

The encryption function is given by:

$$c \leftarrow \mathbf{Enc}_k(m) = m^e \mod n$$

The decryption function by:

$$m \leftarrow \mathbf{Dec}_k(\mathbf{Enc}_k(m))$$

We now have for every ciphertext element $c_i \in \mathcal{C} = m_i^e \mod n$: a public key $k = (e, m)$, we may then derive our private key by choosing an integer $d = (d, n)$ which satisfies $(ed - 1)/\phi n$.

Then every message $m_i \equiv c_i^d \mod n$.

The above example shows that security of RSA depends in some way, that neither p nor q will be approximately $\sqrt{N}$, if this condition turned out to be true then N is called *weakly composite*, or a weak composite.

However, the above condition being true would not be the only way for p and q to create a weak composite N.

We know that not all composites have an equally hard prime decomposition. In fact, not only are there primes of many different forms of prime that will always, (even just sometimes, and nearly always unexpectedly) create composite integers that will be efficiently and easily factorable; we don't even know all the ways or types of primes where a weak composite may occur.

Since we don't know all the forms of primes, or products of primes that will produce easily decomposable numbers, we can't test for them in anyway that assures we've found a reasonable majority.

We want to believe these problems are hard to solve, but despite our best efforts we have no evidence that they really are. For instance, we would like to believe that RSA is *at least as hard to solve* as the integer factorization problem, but we have no proof that this is true.

Even more, didn't we say that the statement *'hard to solve'* meant in the worst-case instance? In reality, many cryptographers have hoped choosing (or not choosing) primes of a certain type would ensure that N would be hard to factor and even when the reasons a particular form of primes to removed from the set were more than reasonable, the system would still produce some other different type of prime which caused the scheme to fail.

So how do we get away with assuming that integer factorization being hard, is at all a reasonable assumption to make? More generally how can we get away with any similar assumption which might be made for a given cryptosystem.

7.2.1 Hardness Assumptions

We define the three standard types of complexity problems: *best-case, average-case*, and *worst-case* known as *hardness assumptions.*

Average-case	**Worst-case**
$P \to Q$	$\sim Q \to \sim P$
For some negligible set of random instances, $\mathcal{P}$ may be easy to solve.	$\mathcal{P}$ is hard to solve, for a set of positive random instances.
'Typical' behavior	*'Rare'* behavior
for some	*for all*

Table 1: *Average vs. Worst Case Comparisons*

Definition 23. *Best-case Refers to the set of problems $\mathcal{P}$ which (uninterestingly enough) have perfect conditions, and outcomes.*

We can interpret the average-case mathematically terms as the quantifier *for some.* The average-case set of instances of a problem are the ones which cryptographic algorithms require by default. They are the problems which are the most familiar, integer factorization, discrete logarithms, and other problems in which we are only able to assume are hard to solve. We do have hope that our hardness assumptions are still reasonable and these lie in two sets of problems for non-deterministic algorithms.

Definition 24. *Average-case For some negligible set of random instances, the problem $\mathcal{P}$ may not be hard to solve.*

The worst-case can be defined as the contrapositive of the average-case, which changes the interpretation in a similar way, as the quantifier *for any*, or *for all*, namely:

Definition 25. *Worst-case A problem $\mathcal{P}$ is hard to solve (*intractible*)*[10] *in the worst-case, for any positve set of random instances.*

The following table compares the differences in definition of average and worst-case hardness assumptions.

Summary 2. Negligible set means that the size of the set is virtually zero.

This implies that for some positive set of random instances we have a probability which is virtually 1 for the complement.

[10]Tractible: (first known usage 1502) meaning *easily managed*. In contrast, intractible (earliest usage 1545) means difficult to manage. In mathematics, refers to a problem which easily lends itself to solution. Computer Science: A problem which can be solved algorithmically, usually in polynomial time.

How do you cite Wikitionary?

The definition of worst-case as discussed in lattice-based cryptography is quite different than the traditional form used in the field of Theoretical Computer Science (TCS). For most applications in cryptography the worst-case was not very useful at all, the worst-case simply meant that there existed hard to solve instances of a problem which applied under the worst-case solvability constraints. In fact, many problems which would seem to be hard to solve in the worst-case instances, would ironically turn out to be easily solvable on the average. This would often be true in cryptography, where secret keys would produce instances with added structure [55].

For the most part, applied cryptographers did not need to worry too much about worst-case instances of problems. One might run a quick analysis of these instances to see if any vulnerabilities could be spotted, and perhaps eliminated, but overall they were ignored (sometimes mocked, particularly by applied students poking at theoretical ones).

8 Quantum vs. Classical Cryptography

The question of if a quantum computer could prove to be more powerful than a classical one was first proposed by Richard Feynmann in 1982. Feynmann first wondered if quantum mechanics might provide a more power computing device than a classical device based on binary type bits. He then proposed the converse of his first question implicitly in 1986: *'By using quantum mechanics can you compute more efficiently than on a classical computer?'* Feynmann, proceeded to formally define quantum Turing machines and quantum bits in order to investigate these questions further.

The first example of a quantum computer is the D-Wave quantum annealing computer. The D-Wave is advertised as *'The world's first commercially available quantum computer'*.

In 2013 Google, NASA Ames, and the Universities Space Research Association collaborated in a purchase of a subclass of the quantum annealing computer called an adiabatic quantum computer.

However, D-Wave systems are not general purpose quantum computers, and were not created with problems such as integer factorization, and solutions to discrete logs in mind. The D-Wave systems were created as special purpose quantum computers to solve optimization problems which involve finding ground states of a classical Ising spin glass.

Update with Google info from Dec

In classical cryptography the problems which serve as mathematical basis for their constructions can all be efficiently [11] solved by quantum algorithms.

The first algorithm which could provide solutions for large integer factorization, and discrete logarithms was proved by Peter Shor in 1997 [**?**]. Solutions for these two mathematical problems effectively destroy the RSA, Diffie-Hellman, and elliptic curve cryptosystems in such a way that a patch such as making the integer much larger in the RSA problem would not be any real or long term solution.

Present quantum computers do not run most quantum algorithms. In particular, they do not run Shors.

[11] Efficient in this context means algorithmically solvable in polynomial time.

	Classical	**Quantum**
Number of Components	n-components	n-components
States	2 states	each state is a point in a 2^n dimensional vector space
Complete State Description	n-bits	$2^n - 1 \in \mathbb{C}$

Table 2: Classical vs. Quantum State Descriptors

Note 2. An 'watershed' announcement is expected on Wednesday, December 8th, 2015 from Google and NASA concerning some quantum breakthrough.

A general purpose quantum computer is one that is capable of carrying out a set of standard quantum operations in any order it is told. [12]

With these developments in mind it would be very poor risk management to delay development of quantum resistant cryptosystems and assume the longer time frame. It is quite plausible that not all developments are within public knowledge.

8.1 Basic Definitions in Quantum Computing

Below we give the basic definitions used in quantum computing which are relevant to cryptographic schemes.

Classical computers use binary bits as their fundamental unit, the value of a bit can only ever be 1 or 0. Binary bits are operated upon pair wise by Boolean logical connectives such as $\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}$. Sequences of binary bits are manipulated by Boolean logic gates which operate in succession until an end state or computation has been completed.

In contrast, quantum computers have as fundamental units, quantum bits or *qubits.* Sequences of qubits are operated upon by quantum logic gates, which simulate particular laws of quantum mechanics such as *superposition* and *entanglement.* The following table compares the states and descriptions of classical and quantum computers.

[12]There do not at this point in time exist any general purpose quantum computers, However this may not be true come Wednesday.

Classical Resources	Quantum Resources
Time	Time
Space	Space
	Precision

Table 3: Classical vs. Quantum Resource Comparisons

Notation 1. For each of the 2^n possible positions of the components in a quantum computer, there exists a basis state of the vector space represented as:

$$|011\dots0\rangle$$

where $|x\rangle$ denotes a pure quantum state. (ket notation)
Property 1. The Hilbert space associated with the quantum system is the complex vector space with the 2^n states represented as basis vectors.
Property 2. The state of the system at any time t is represented by a unit-length vector in Hilbert Space.
Notation 2. Superposition of the state is represented by:

$$\sum_{i=0}^{2^n-1} a_i S_i : a_i \in \mathbb{C}$$

these are the amplitudes,

$$\sum_{i=0} |a_i| = 1,$$

and each S_i is a basis vector of the Hilbert space.
Assumption 19. Since multiplying the state vector by a unit-length complex phase makes no change in the behavior of the state, it only requires $2^n - 1$ element of $\mathbb{C}$ to represent the complete state of the system.
Property 3. A quantum computer must be able to make changes at will to the quantum states of objects. Inherently, the precision cannot be perfectly measured, i.e. the measurement contains some degree of error (has an error term). (That is it has some amount of imprecision).
Property 4. If the device is measured with respect to this basis at any step the probability of seeing the basis state: $|S_i\rangle$ is $|a_i|^2$.
Remark. However, the act of measuring projects the state to the observed basis vector $|S_i\rangle$. Therefore, looking at the machine during

Constant	**Non-Constant**
Constant degree of error means that the measure of precision does not depend on the size of the input	Non-constant degree of error means that for the input size the precision is permitted to grow in poly-log time
	Meaning if the number of bits of precision has a log input size for the growth rate then a quantum computer is more powerful than the classical
	The power result is due to the fact that, allowing a non-constant degree of precision does not *'appear'* to confer any additional computational power
	Although allowing exponential growth may
At the time of [[**?**]] it was unknown *HOW* to compute any functions in polynomial time on a quantum computer that could not also be computed in polynomial time on a classical computer using a random number generator	

Table 4: Constant and Non-Constant Precision Comparisons

computation invalidates the remainder of the computation. (*Schrodinger's cat*)

Quantum computers have analogous forms of the oracles, Turing machines, and other devices used in computability theory for classical systems with slight modifications due to their nature. For instance, the description of a quantum Turing machine must include an error term which to account for the additional possible states of a qubit.
Note 3. If a quantum Turing machine is allowed a small probability of error then the quantum gates and quantum Turing machines can compute the same function in polynomial time. This implies that the class of functions which are computable in polynomial time is robust, with a small degree of error which is not dependent on the exact architecture of the quantum computer.

8.2 Shor's Algorithm

In 1997, Peter Shor described an algorithm that could easily decompose large integers into their prime factors in polynomial time with a high probability of success. This decomposition breaks crypto-

Quantum Class	**Classical Class Analog**
BQP	BPP
Bounded Error Probability	
Quantum Polynomial Time	

Table 5: Class Comparisons Between Quantum and Classical Systems

graphic algorithms such as RSA. Shor also described an algorithm which solves the discrete logarithm problem in $\mathbb{F}_p^*$. There now exist variants of Shor's solution for the discrete log which can solve the elliptic curve discrete logarithm problem used in *eliptic curve cryptography* as well, in polynomial time.

Recall the Fourier transform takes a complex function of time (a signal) and breaks it up into its components. The discrete Fourier transform operates similarly on samples which have been taken in discrete units of time; it uses the Poisson summation formula in order to produce a frequency which represents the continuous Fourier transform of the original function.

The key to Shor's algorithm makes use of a quantum variant of the Fast Fourier Transform, this transformation can be found in polynomial time. The next step is to use the transformation on a particular superposition of quantum states and measure the system. The probability of observing a state is large enough if there exists a rational number $\frac{s}{t}$ which satisfies $\left|\frac{p}{q} - \frac{s}{r} < \frac{1}{2q}\right|$, where r is the order of a number $a \mod n$ such that $0 < a < q$, and $|a\rangle$ is a state on which the transform operates.

Notably, Shor's algorithm is capable of solving all current methods typically used in classical cryptography with one exception. Shor's algorithm nor any other quantum or classical algorithm has yet to produce a solution for a worst-case problem defined over lattices. This of course does not imply no such solution can be found.

Mathematical Problem	**Cryptographic Scheme**
Integer Factorization	RSA
Discrete Logarithm	Diffie-Hellman Protocol
	The Digital Signature Algorithm (DSA)
	ElGamal (PGP)
Elliptic Curve Discrete Logarithm	Elliptic Curve Diffie-Hellman (ECDH)
	Elliptic Curve Digital Signature Algorithm

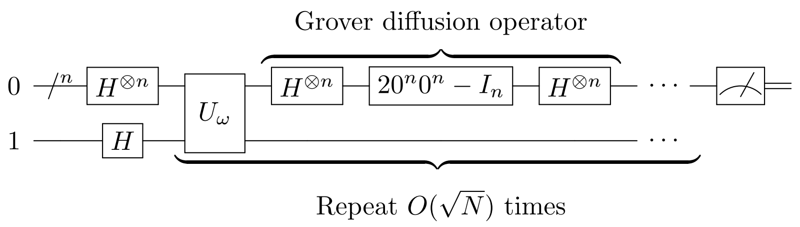
Table 6: Classical Algorithms Susceptible to Quantum Attack

Table 7: Efffect of Quantum Computing on Current Cryptosystems

Survive	**Die**
Merkle hash-tree PKC signature Scheme	DSA
McEliece hidden-Goppa-code PKC Encryption Scheme	ECDSA
Lattice-based Cryptography (NTRU)	EDD (n general)
Multivariate Quadratic Equations (Patarin HFE$^{v-}PKCSigscheme$)	HECC (in general)
Secret Key Crypto Rijndael (AES)	Buchmann-Williams
	Class groups in general

The following table shows a few examples of cryptosystems which can be solved using Shor's algorithm, we refer to them as quantum unsafe, likewise if a problem seems to provide some degree of security against attack by a general purpose quantum computer we refer to it as *quantum safe*.

Any sufficiently large general purpose quantum computer can 'easily' solve the mathematical problems below using Shor's or another quantum algorithm.



8.3 Grover's Algorithm

Grovers algorithm is a probabilistic quantum search algorithm which is able to find the unique input of ablack box function f, which maps to a unique output in $\mathcal{O}(N^{1/2})$ calculations, where $N = |Dom(f)|$. It was shown to be optimal by [22], with only $\mathcal{O}(N^{1/2})$ minimal number of evaluations required to find a solution. Despite that Grovers is a probabilistic algorithm the number of evaluations expected before a solution is found is constant and not dependent upon N.

Grover's only speeds up the computation of a black box function only quadratically. For large enough domains, N is fast enough to allow brute force solutions of 128 bit symmetric keys in approximately 2^{64} calculations, 256 bit in 2^{128}.

Since Grovers is a search algorithm (usually defined for databases) it is possible to consider the input and output values of f as corresponding database entries and 'search for one or more x values to satisfy a given y value. That is, we can think of Grovers as performing an inversion on $y = f(x)$ and returning the correct x.

[23]

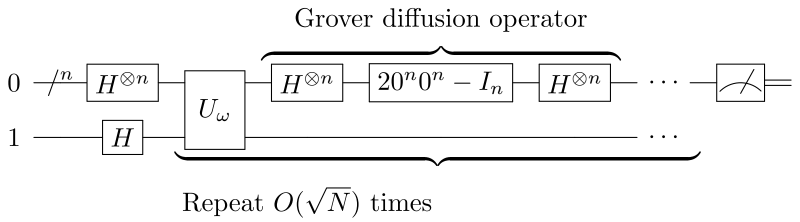


Figure 2: Grovers Algorithm Quantum Circuit

8.4 Grover's Algorithm

Grovers algorithm is a probabilistic quantum search algorithm which is able to find the unique input of ablack box function f, which maps to a unique output in $\mathcal{O}(N^{1/2})$ calculations, where $N = |Dom(f)|$. It was shown to be optimal by [22], with only $\mathcal{O}(N^{1/2})$ minimal number of evaluations required to find a solution. Despite that Grovers is a probabilistic algorithm the number of evaluations expected before a solution is found is constant and not dependent upon N.

Grover's only speeds up the computation of a black box function only quadratically. For large enough domains, N is fast enough to allow brute force solutions of 128 bit symmetric keys in approximately 2^{64} calculations, 256 bit in 2^{128}.

Since Grovers is a search algorithm (usually defined for databases) it is possible to consider the input and output values of f as corresponding database entries and 'search for one or more x values to satisfy a given y value. That is, we can think of Grovers as performing an inversion on $y = f(x)$ and returning the correct x.

[23]

9 Basic Lattice Concepts

Definition 26. *Lattice Point A lattice point is a point in Euclidean space which sits at the intersection of one or more edges in the grid of a lattice. In particular, we are concerned with discrete lattice points which are isolated integer values around which there is a neighborhood where no other such points can be found.*
Definition 27. *Lattice Let* $\mathbb{R}^n$ *represent the n-dimensional Euclidean space. A lattice in* $\mathbb{R}^n$*, denoted* Λ *is the set of all integer linear combinations of independent basis vectors* $\mathbf{b}_m \in \mathbb{R}^n : m \leq n$ *which satisfy:*

$$\Lambda(\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}) = \left\{ \sum_{i=1}^{m} x_i \mathbf{b_i} : x_i \in \mathbb{Z}, \mathbf{b_i} \in \mathbb{R}^n \right\}$$

Definition 28. *Integer Lattice An integer lattice* $\mathbb{Z}^n$ *is an n-dimensional lattice in* $\mathbb{R}^n$ *whose lattice points are all integers.*

We say that n is the *dimension* of the lattice, and likewise m is the *rank*. When the number of integer points (intersections, or coordinates) of the lattice is equal to the number of basis vectors, then the dimension is equal to the rank and we say the lattice is *full-rank*.
Theorem 2. *A lattice* Λ *is full rank if and only if its linear span* $span(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{R}\}$ *is equal to to all of* $\mathbb{R}^n$*.*

The m vectors $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}$ are called the *basis for the lattice*, which can be represented by a matrix $\mathbf{B} = [\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}]\ \mathbb{R}^{n \times m}$. Written as column vectors we have the compact representation: $\Lambda(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n$, under matrix/vector multiplication.

9.1 Short Integer Solution (SIS)

The Short Integer Solution Problem (SIS), was the first case where a connection between the average and worst-case instances of a problem was discovered. The connection was discovered by Mikolos Ajtai in 1996 [11] and serves as the basis for a variety of problems throughout the various applications of cryptography. Examples include collision-resistant hash functions, authentication and identification schemes, digital signatures, and countless others with the exception of any application to a public-key scheme.

The problem posed by SIS is as follows:
Question 1. Short Integer Solution Given a finite number of elements drawn at uniformly and randomly from a certain finite additive subgroup, find a sufficiently ‘short’ nonzero (nontrivial) linear combination of these elements which can be summed to zero.

Formally, SIS is a parametrization of integers n and q which define the subgroup $\mathbb{Z}_q^n$ for a real number β and a some number m of group elements. Where $n \leq 100$ is the hardness parameter, and $q > \beta$ is some small polynomial dependent on n.
Definition 29. *Short Integer Solution Given m number of uniform, random vectors $a_i \in \mathbb{Z}_q^n$, which are column vectors of a matrix $M \in \mathbb{Z}_q^{n\times m}$, find a nonzero vector $z \in \mathbb{Z}^m$ with norm $\| \leq \beta$ such that*

$$f_M(z) := Mz = \sum_i a_i z_i = 0 \in \mathbb{Z}_q^n$$

Note 4. Observe that the constraint $\|z\|$, is necessary or the problem can be easily solved by Gaussian elimination. Likewise if $q \nleq \beta$ then we have a trivial solution $z = \{q, 0, \ldots, 0\} \in \mathbb{Z}^m$, since any such solution for M can be converted into any other solution $M|M'bypaddingthesolutionwithzeros, whichhastheuselessresultthatthenormisneveralteredi$

9.2 Learning with Parity (LWP)

An algorithm trained to solve the Learning with Parity problem is given the following parameters:

[13]If this reminds you at all of the caterpillar from Alice in Wonderland here, that would be entirely appropriate.

- Samples $(x, f(x))$, where the samples are generated by some distribution over the input.
- For the function f the algorithm has the *assurance* that f can compute the parity of bits at one or more fixed locations.

If the algorithm solves the LWP problem it has successfully guessed the original function f from the given samples.

The above version of LWP is easy to solve using Gaussian elimination provided the algorithm is given enough samples, and the distribution from which the samples are drawn is not overly skewed.

9.3 Noisy Learning with Parity

Noisy Learning with Parity (NLWP) deviates slightly from the constraints given above significantly altering the solvability of the problem.

The algorithm's parameters are adjusted as follows:
Condition 1. The algorithm is provided with samples (x, y), such that $y = 1 - f(x)$ has only a small probability of knowing the parity of bits at some locations.
Condition 2. These samples are permitted to be noisy, that is they may contain errors.

The adjustments to the LWP problem change the expected outcome from one which is relatively easy to solve to one which is conjectured to be hard to solve.

9.4 Learning with Errors (LWE)

Learning with Errors is a lattice problem based upon the above machine learning problem of Noisy Learning with Parity (NLWP).

LWE was described and shown to be hard in the worst-case by Oded Regev in 2009 [62]. LWE, has a quantum reduction from the Shortest Vector Problem (SVP) which is known to be NP-hard. LWE forms the basis of security for many other algorithms which gain SVP as basis for their own security.

The definition of the Learning With Errors problem asks for a way to distinguish between random linear equations, which have been perturbed by the addition of some small amount of noise. Regev proved LWE is *as hard to solve* as certain worst-case lattice problems.

Lemma 1. *Regev's Main Lemma There is an algorithm which takes a basis B of an n-dimensional lattice $\Lambda = \Lambda(B)$, the parameter $r >> \frac{q}{lambda_1(\Lambda)}$, and a point vector $x \in \mathbb{R}^n$ where the distribution $dist(x, \lambda) < \frac{\alpha q}{\sqrt{2r}}$, are taken as input and where α is the noise parameter, n is the security parameter, and q is the modulus.*

The $dist(x, \lambda) < \frac{\alpha q}{\sqrt{2r}}$ has access to two oracles: one oracle is not related to the input lattice and solves $LWE[n, \alpha, q]$

The other, uses a continuous distribution D_r, where r is the standard deviation. And $D_{\Lambda,r}$ is a discrete distribution over the lattice, such that all vectors $z \in \Lambda$ has a probability mass proprtionate to $D_r(z)$. Then the second oracle is a sampler which is specific to a given lattice.

For the inputs, and oracles that are given above. LWE efficiently finds the unique point $v \in \Lambda$ which is the closest point to x. Further, LWE finds this point with a large probability.

However, despite that LWE has allowed for many new, and secure constructions it suffers from a large quadratic overhead cost (it is inefficient). For this reason the hypothesis by Chris Peikert was made that perhaps if the lattices were permitted to have some additional structure the new applications which based their security on the worst-case quantum hardness of LWE could be made to be efficient, perhaps even competative with current classical schemes.

9.5 Ring-LWE (RLWE)

Peikert's conjecture concerning the addition of structure to the general lattices of LWE, indeed produced a set of lattices defined over rings which allowed the new applications a much greater range of efficiency. Provided that one made the assumption that ring-based lattices retained the worst-case quantum hardness afforded to LWE.

9.6 IR-LWE

In cases where the polynomial $\Phi(x)$ is a cyclotomic polynomial, the difficulty of solving the search version of RLWE problem is equivalent to finding some short vector vector in an ideal lattice formed from elements of $Z[x]/\Phi(x)$ represented as integer vectors.

This problem is commonly known as the *Approximate Shortest Vector Problem* (α-SVP), where the problem is to find some vector in the lattice with length that is smaller than a multiple of α and the *shortest vector* in the lattice.

10 Post-quantum Cryptography

Post-quantum cryptography is field of cryptologic research that is concerned with the task of using classically computable cryptosystems to defend against quantum attacks[14].

[14]Not to be confused with quantum cryptography which is the use of quantum computers and quantum algorithms to create and analyze cryptographic schemes

11 Lattice-based Cryptography (LBC)

Lattice-based cryptosystems are a subset of Post-quantum cryptographic schemes based over integer and algebraic lattices.

In nearly every *classical* cryptographic construction of primitives proceeds as follows:

> Generate a specific instance of a problem and it's solution in the complexity class **NP** such that it is reasonable to conclude the problem is 'hard to solve' for some reason; where 'hard to solve' is meant in the worst-case.

In 1996 Ajtai described the first average to worst-case hardness connections.

12 Ideal Lattice-based Cryptography

The additional application areas and ease of implementation due to ideal ring structure is convenient, the choice of ring its self is as important as the problem it is based upon. An advantage of Ring-LWE is its much smaller set of keys; the ring allows for more efficient and larger ranges in application areas.

Ideal Ring-LWE allows an even broader range of applications and efficiency than RLWE due to the additional structure provided by the principal ideals.

12.1 Ideal Ring Lattice Cryptosystems

Gentry's Scheme

[?] Was the first successful construction fully homomorphic encryption scheme In 2009 Craig Gentry constructed the first cryptographic system capable of evaluating arbitrary operations. The scheme was constructed over a principal ideal lattice [?] uses a bootstrap method to prove its security.

12.2 Principal Ideal Lattice Insecurities

The cyclotomic field of characteristic two infers so many convenient structures it is often chosen without consideration to any other ring [?].

In 2010 at the Public Key Cryptography Conference two researchers Smart and Veracauteren [5] introduced another ideal lattice scheme promising fully homomorphic encryption and quantum assumption hardness proofs. In late 2014 GCHQ (Government Communications Head Quarters) announced that it had been working on such a scheme, nearly identical to [5] and claimed that not only was it easy to solve using a quantum computer but was subexponential for classical computer systems as well. The attack did not directly apply to most constructions of lattice-based schemes only a handful who had used ideal lattices in the cyclotomic field of characteristic two, where the private keys were represented by principal ideals which were short generators.

A team of researchers Ronald Cramer, Leo Ducas, Chris Peikert, and Oded Regev [**?**] who had prior to CESG announcement strongly encouraged new constructions in other fields quickly proved CESGs claim. Moreover, they proved that in cyclotomic fields with dimensions of prime power order which guaranteed existence of a short generator the lattice could always be efficiently decoded revealing the private key.

13 Fully Homomorphic Encryption

An added benefit of Lattice-based schemes is the potential of providing cryptosystems equiped with Fully Homomorphic Encryption (FHE).

Fully Homomorphic Encryption is best informally explained by a usecase in the context of web searches.

Example 9. Suppose Bob query's a search engine for content which may have a sensitive context. Currently Bob's available set of protective actions consist of a limited, complicated[15], and non-optimal (or fool proof) proceedure for minimizing risk while searching. Should his search engine decide it would like to analyze, sell, or make use of Bob's search data, Bob has no choice in how, when, or why his data will be used.

However, if Bob had a computer which operated not on ordinary binary bits but encrypted bits instead, he might have some small chance that his sensitive data remained secure. This is one example of a possible use of Fully Homomorphic Encryption. If Bob was able to use an FHE system his footprint would be much simpler. Bob only needs to encrypt his query, send it to the search engine, and recieve his results. The search engine only would see something was searched for, and some result was produced. It would not see what Bob looked for, at which results he decided to look more closely, and which results he saved or discarded.

In general, the problem which FHE solves is only slightly different than that of Public-key. We want to take our encrypted data, send it to an unauthorized party for processing, and receive the same results we would get if we had never encrypted at all.

The ability to send and process encrypted data as if it had not been encrypted at all is made possible by the use of simultaneous homomorphisms.

Definition 30. *Homorphism (informal) Informally, a homomorphism is a mapping between two objects which preserves some spec-*

[15]to erase his browsing history, use a vpn (I am not including Tor or anonymizing services in this example), remove all scripts and applets, enable anonymous browsing mode in the browser of his choice, cross his fingers and hope the search engine or his ISP will not find him interesting (additionally, hope that his anonymous browser has patched it's 'super cookie' tendencies).

ified structure. Not all structures of an object must be preserved under this mapping.

Definition 31. *Homomorphism of Rings*

Let $\mathcal{R}$ and $\mathcal{S}$ be rings, then $f : \mathcal{R} \to \mathcal{S}$ is a homomorphism of rings if for all $a, b \in \mathcal{R}$ the following conditions hold:

1. $f(a + b) = f(a) + f(b)$,
2. $f(ab) = f(a)f(b)$, *and*
3. *For an identity element e we have that $f(e_{\mathcal{R}}) = e_{\mathcal{S}}$.*

Note 5. The definition of a ring implies the existence of additive inverses and additive identity so these conditions need not be explicitly specified in the above definition for a homomorphism.[16]

You might ask:

Why is FHE 'special'? If we've had homomorphisms in our crypto for ages!

13.1 Homomorphic Properties of Cryptographic Schemes

It is true that many classical public key cryptography methods have a homomorphic property, this property says if add two things together then their sum contains information about those two things. That is, they use successive operations of addition, multiplication, etc. to create ciphertext.

The keyword being *successive* operations.

In the context of cryptography (since 2009) we call these schemes *Partially Homomorphic*.

Unpadded RSA has this particular homomorphic property. That is, if we take two RSA messages (ciphertext) which have been encrypted with unpadded RSA, and we multiply them together, when we decrypted them we will find that this product contains the two original ciphertexts as subgroups, the ciphertexts have been embedded in their product.

The embedding the homomorphic property made possible in unpadded RSA is of course why we do not use unpadded RSA. We

[16]See Appendix G for basics of rings.

need two more security definitions to understand what FHE is, as well as what it attempts to provide.

13.2 Malleability Avoidance

In most classical cryptosystems great care has been taken in avoiding malleability, a property which comes as a byproduct of homomorphic functions. Consider the following definitions and examples which show why this avoidance was necessary.

Definition 32. *Malleable Encryption*

If c' is a similar ciphertext for c. That means if c can be transformed into some other c' such that whenever we decrypt both c and c' we recieve either the same or a closely related plaintext, $f^{-1}(c) = f(m) = f^{-1}(c')$.

That is, a cryptosystem is called malleable *if when we know f, it is posible to take some ciphertext c and modify it to a new c' then we have modified the message m to one of our own choosing, without having to decrypt m in the first place.*

Note 6. An important property to notice in the definition for malleability is that a cryptosystem (such as unpadded RSA) is still considered to be secure from CPA, CCAI, and semantically. The adversary did not learn any information about the plaintext of the message, they did not need to.

When might this property become a liability for a scheme?

Example 10. Suppose Alice sends a message to Bob telling him to meet her at the deli at five oclock. If Eve would prefer Bob be somewhere else or simply not be at the deli at the same time as Alice she would only need to alter the ciphertext in a way that changed the time or place. Eve wouldn't need to know the contents of the message to choose either.

Another more practical example would be in terms of bank transfers.

Example 11. Suppose Bob sends Alice a bank transfer for 50\$. Eve only needs to intercept a bank transfer until she aquires a ciphertext she can manipulate. Eve may then (if the banks cryptosystem is malleable), transform the ciphertext to decrypt into an amount she would like, say 5000\$, then redirect the money to an account of her own choosing.

Eve does not need to know how much the orginal transfer was made out for, who it was directed to, or who sent it to begin with; she only needs access to a manipulatable ciphertext.

We now can clearly see why cryptographers might want to go out of their way to avoid this particular property. This unfortunate property enabled by homomorphisms leads us to the following definition:
Definition 33. *Plaintext Awareness*

A given cryptosystem is considered to be plaintext aware *if it is infeasible for an adversary to create a valid ciphertext without knowing the corresponding plaintext. [**?**]*

The examples above hint at when this definition might imply the security of a cryptosystem, but consider the following senario.

If Bob takes his message and manually using a cesar cipher transforms his plaintext into ciphertext he is aware of not only the content of his message, but its correspondence the ciphertext (he knows the shift). However, many modern cryptosystems do not require the sender to know anything at all about the plaintext.

To use the example of unpadded RSA again; in RSA the plaintext and corresponding ciphertext are both taken modulo n; so unpadded RSA is not plaintext aware. Meaning, a valid ciphertext can be found simply by picking some number modulo n.

On the other hand, if a cryptosystem is plaintext aware. Then the cryptosystem is both semantically secure and secure from any chosen-ciphertext attacks (CCAI and CCAII). Plaintext awareness is a very strong condition for the security of a given cryptosystem since it does not permit the adversary to find any information about a plaintext from its encrypted form they did not already know.

In fact, any cryptosystem which is secure against the CCAII attack model is by definition a *non-malleable* cryptosystem.

13.3 Malleable on Purpose?

Not all cryptosystems avoid malleability; some cryptosystems exploit the property in performing their primary functions. Indeed, Fully Homomorphic Encryption schemes exploit malleability by using the fact that such systems permit the manipulation of ciphertexts without ever knowing the contents of the plaintext.

The homomorphisms used by Fully Homomorphic Encryption schemes allow two distinct operations to be performed not in succession but in tandem. Meaning, FHE schemes are not limited to a single operation or more operations which are performed sucessively, they can simultaneously compute arbitrary operations.

By design FHE schemes are malleable; in this case the property allows operations to be performed on ciphertext as if it were plaintext returning the same result as if the operations had been performed on plaintext.

13.4 Operations on Arbitrary Circuits

All computers operate by using trillions upon trillions of boolean circuits. Boolean circuits are special cases of propositional formulas, truth tables, where the only output ever given by the system is either yes or no; in computer terms zero or one.

Boolean circuits have three basic functions from which the rest can be derived, they are AND, OR, and NOT. You can combine these two of functions to get a total set of sixteen different boolean functions. Engineers have been using these functions for ages, but cryptographers were left with functions defined for either single operations, or multiple operations which could only be performed in succession.

The concept of Fully Homomorphic Encryption, first titled *privacy homomorphisms*, was proposed by Rivest in 1978[64] following the release of RSA. Though many attempts were made to create such a scheme not a single construction was successful until 2009 when Craig Gentry described a cryptosystem defined over ideal lattices in his PhD Thesis[33].

Before we jump into a description of Gentry's scheme we need to understand what it means for a cryptosystem to be capable of evaluating two or more operations simultaneously.

As stated above all electronic devices perform operations on two or more Boolean functions on the set $\{0, 1\}$, representing the state of

a circuit being ‘on’ or ‘off’. However, in order to avoid malleability cryptographic schemes usually removed the homomorphic properties endowed by their functions, this meant simultaneous evaluation on two or more operations was not feasible to construct without a significant loss in security.
Note 7. FHE schemes are malleable by design; therefore, while they do meet the lower bound of being semantically secure, the semantic property for security is not sufficient condition to shield most modern cryptographic schemes from attack; this bound is provided by the adaptive chosen-ciphertext attack (CCAII) model. FHE schemes by definition are only capable of providing at most security in the CCAI attack model.
Note 8. Cryptographers for the most part during the past three decades had to remain content with creating schemes specifically designed to ingest particular types of input and perform a specialized type of function. But Rivest had shown the dream of cryptographers was on par with that of engineers, to create systems which could operate over arbitrary circuits as any other device, with a key exception. The exception being that fundamental units these circuits operated on were not ordinary bits, but cryptographic bits.

A computer which could evaluate arbitrary circuits over encrypted bits can perform any function of one which operates on standard bits. For example, if you were to manage your accounts on a typical computer you would need to know that the connection to your bank, storage method used of your records, the personell allowed to perform additional calculations, etc all met some minimum level of security. If you wanted to manage your accounts with a system operating on cryptographic bits you could simply encrypt your records and send them off to be processed

13.5 Gentry’s Fully Homomorphic Encryption Scheme

13.5.1 Overview of Gentry’s Method

1. Construct an encryption scheme which can evaluate arbitrary circuits.
2. Describe a public key encryption system using ideal lattices; we call this scheme a Semi Homomorphic Encryption scheme

(SHE).

3. Modify the SHE scheme constructed in step 2 by reducing the depth where the scheme evaluates a circuit correctly so that it will be logarithmic with respect to the dimension of the lattice for its decryption circuit.

Note 9. On Step 1: For step one it is sufficient to construct an encryption scheme which is capable of evaluating slight augmentations of its own decryption circuit.

Note 10. On Step 2: To describe a SHE scheme defined over ideal lattices note that:

- Algorithms for lattice-based schemes typically have low circuit complexity, which is dominated by an inner product operation in the **NC** complexity class.
- Ideal lattices are the ideals of a polynomial ring with a lattice representation; this homomorphism provides the additive and multiplicative properties needed in order to evaluate arbitrary circuits.

Note 11. On Step 3:

Reducing the depth of the decryption circuit, takes the scheme to a point where bootstrapping may now occur, without reducing the depth the scheme can evaluate circuits.

The bootstrap is accomplished by allowing the encryption function to start the decryption process which reduces the work of the decryption function.

13.6 FHE Pseudocode

We start with a SHE scheme, and two functions $f_\wedge$ and $f_\vee$, for $x, y \in \{0,1\}$.

Define the encryption function homomorphically:

$$Enc(x \vee y) = f_\vee(Enc(x), Enc(y))$$

And,

$$Enc(x \wedge y) = f_{\wedge}(Enc(x), Enc(y))$$

The SHE cryptosystem consists of probilistic polynomial time functions, $Gen, Enc, Dec, Eval$.

The key pair contains (p_k, s_k, ev_k), where p_k is a public encryption key, s_k is the usual private key decryption key, and where ev_k is a public evaluation key.

Key Generation proceeds as follows:

$$(p_k, s_k, ev_k) \leftarrow SHE.Gen(1^{\lambda})$$

where λ is a security parameter.

Let c be a ciphertext, $\mu \in \{0,1\}^1$, and r is a randomizer, then we make the first encryption in the following way:

$$c \leftarrow SHE.Enc_{p_k}(\mu, r)$$

The first decryption

14 Counterexamples Win Out

The security of a cryptosystem must necessarily take higher precedence over the desire for efficiency. Neither condition can be ignored but the purpose of the scheme is to prevent the observation of data by unauthorized parties; it is possible to phrase this objective sufficiently but not necessarily. A slow system, even if communication at a snails pace is less than optimal or even usefull, will still fullfill the purpose in create a secure communication system.

However, if we allow a cryptographic scheme to be efficient but we do not provide a lower bound for it's security there is no reason for expending effort to create, use, or acknowledge the scheme at all. The scheme has become a false statement. There are many instances in which such a choice was made in modern cryptography. In Particular, a recent example comes from a set of cryptographic schemes used in ideal lattice-based cryptography. In this instance, a great many researchers took for granted the level of security lattice-based schemes were conjectured to impart, and turned their focus entirely to the task of advancing the efficiency of their schemes instead. These types of constructions on ideal rings soon became the largest subset of all possible lattice problems not just those defined over rings. Researchers increasingly assumed without proof that these problems would prove to be hard without attempting to show that it was or was not the case more, assuming their systems would prove to be as secure as schemes defined over more general lattices, but leaving the task to someone else.

In mathematics the, the best possible refutation of a claim something does not or cannot exist or have a certain property is a counterexample. Counterexamples need not be proven for more than one case, and in applied areas these examples often come in the form of use case. Cryptography is no exception to this rule. Particularly since it requires many assumptions to even begin a new construction.

The faith of the ideal lattice cryptographers in the strength of their efficient constructions is not a new mistake in the history of cryptography. This assumption of strength is famously mirrored in the arrogance of engineers of the German Enigma ciphers of World War II, faithfully relying on their calculations of theoretical strength, and refusing to compute any possible realworld value for the system.

There is no better such example in modern lattice-based cryptography then the Soliloquy problem; an unexpected announcement by British researchers in the Communications-Electronics Security Group (CESG) (the information assurance division of the GCHQ). This informal report dubbed *Soliloquy: A Cautionary Tale* [28], brought on a slew of uncharacteristically emotional responses in the cryptographic community, a mixture of shock, speculation, [17]

The announcement by CESG, vaguely outlined a cryptosystem given over ideal lattices of characteristic two, a characteristic much overused despite the warnings of more experienced researchers [29] that efforts in other fields were necessary in order to guard against the possibility that a particular field characteristic was shown to exhibit such a vulnerability (and many other very good reasons).

[17] As well as a few cases of well-earned, smug *'I told you so'* sentiments. As for researchers working towards efficient constructions of similar schemes there was a backlash of reactions ranging the spectrum of in one direction, unsurprised acceptance and in the opposite direction, rage and denial.

15 Soliloquy Construction

The following definition for the Soliloquy cryptosystem was compiled using the information from [28], [**?**], and [**?**].
Definition 34. *Key Pair*

Let (p_k, s_k)*, be the public and private key pair for the system such that:*

Let ζ *be a primitive* n^{th} *root of unity.*

We define $\mathbb{K}$ *to be the number field* $\mathbb{Q}(\zeta_n) : n > 2$*, and define* $\mathcal{R}$ *to be the ring of integers* $\mathbb{Z}[\zeta] \in K$*.*

The public-key, $p_k \in \mathbb{P}$*, such that* $p_k \in \mathbb{K} : n > 2$ *is a large prime* p*, with a bit size between* $3000 \leq |p_k| \leq 10,000$*, dependent on level of required security.*

The secret-key $s_k = p_k j = \theta$ *for some constant* j*, where* $s_k = \theta = |p|$*, is an algebraic integer.*

Let Λ be a cyclic lattice related to θ, such that the principal prime ideal of Λ is generated by θ.
Property 5. s_k

The secret-key s_k enables efficient, decryption if the vectors match a particular form. However, recovery of s_k directly from p is a known hard classical problem for basis-reduction.
Note 12. The Soliloquy scheme can be thought of as a variant of the GGH cryptosystem. In particular, Soliloquy uses GGH's lattice.

Select α as a candidate for $p_k = \sum_{i=1}^{n}(\alpha_i \zeta_i) \in \mathcal{O}$, such that the coefficients α_i are taken from a discrete Gaussian distribution with a mean of zero. i.e. The coefficients are relatively small.

Let p be the norm of α.

The conditions under which an ordered pair, (α, p) can be used as a set of Soliloquy keys (pk, sk), are

- $p \in \mathbb{P}$ and
- $c = 2^{\frac{p-1}{n}} \not\equiv 1(modp)$.
- Then the quantity c occurs with probability $1 - \frac{1}{n}$.

Using these conditions we guarantee that we have at least one principal ideal of the form $\mathcal{P} = p\mathcal{O} + (\zeta - c)\mathcal{O}$, where a prime $p \equiv 1(modn)$ is the principal ideal p_O for a particularly chosen $c = 2^{\frac{(p-1)}{n}}$.

If p_O is a product of prime ideals $\mathcal{P}_i$ with representation $\mathcal{P}_i = p\mathcal{O} + (\zeta - c_i)\mathcal{O}$ where the c_i are the non-trivial n_{th} of unity mod p.

Since the Galois group $Gal(K/\mathbb{Q})$ gives a permutation of prime ideals $\mathcal{P}_i$, then by a reordering of the coefficients a_i, (i.e. taking some Galois conjugate of α), we can ensure $\alpha\mathcal{O} = \mathcal{P}$.

If $p\mathcal{O}$ is a product of prime ideals $\mathcal{P}_i$ with representation:

$$\mathcal{P}_i = p\mathcal{O} + (\zeta - c_i)\mathcal{O},$$

where the c_i are the non-trivial n^{th} roots of unity mod p.

Since the Galois group $Gal(K/\mathbb{Q})$, gives a permutation of prime ideals $\mathcal{P}_i$, by a reordering of the coefficients a_i,

(*i.e. taking some Galois conjugate of* α), we can ensure $\alpha\mathcal{O} = \mathcal{P}$.

We have obtained a natural homomorphism

$$\psi : \mathcal{O} \to \mathcal{O}/\mathcal{P} \simeq \mathbb{F}_p$$

such that:

$$\psi(\epsilon) = \psi\left(\sum_{i=1}^{n} e_i c^i\right) = \sum_{i=0}^{n} e_i c^i \mod p = z.$$

15.1 Encryption & Decryption

15.1.1 Encryption

The encryption function is:

$$\psi(\epsilon) = \psi\left(\sum_{i=1}^{n} e_i c^i\right) = \sum_{i=0}^{n} e_i c^i \mod p = z.$$

The ephemeral[18] key ϵ is sampled from the ring of integers $\mathcal{O}$, where

[18]Emphemeral keys are key which are (usually) generated each time the key generation process is executed.

the coefficients are sampled with a mean value of zero, the encrypted output z, is a positive rational number.

15.1.2 Decryption

To decrypt ϵ, we use the closest integer function $\epsilon = \lceil z \rfloor z\alpha^{-1} \cdot \alpha$, which gives ϵ small enough that $\lceil \epsilon \cdot \alpha^{-1} \rfloor = 0$.

15.2 Cramer et al. Classical Solution

Look up these references and sub them in.

'Indeed, power-of-two cyclotomics have become the dominant and preferred class of rings in almost all recent ring-based cryptographic schemes (e.g., LMPR08, LM08, Lyu09, Gen09b, Gen10, LPR10, SS11, BV11b, BGV12, GHS12a, GHS12b, Lyu12, BPR12, MP12, GLP12, GHPS12), often to the exclusion of all other rings.' [29]

16 Appendix A: Gauss' Circle Problem

Question 2. Given a circle $\mathcal{R}^2$ with radius $r \geq 0$ centered at the origin $\mathcal{O}$, how many integer lattice points (m, n) can you find in the circle?

Since we can express a circle in terms of Euclidean coordinates as:

$$x^2 + y^2 = r^2.$$

We can state the question in terms of our lattice points (m, n) as:

$$m^2 + n^2 \leq r^2.$$

Approximate Solution

The area of the inside of a circle is approximately $A \approx \pi r^2$.

Which gives us an approximate solution in terms of r for Gauss' circle problem (denoted $N(r)$) of:

$$N(r) = \pi r^2 + E(r),$$

where $E(r)$ is some error term given by the radius.

Exact Solutions

Two forms for an exact solution for the circle problem can be given in terms of sums.

The first is a bit ugly:
Solution 1.

$$N(r) = 1 + 4\sum_{i=0}^{\infty}\left(\left\lfloor\frac{r^2}{4i+1}\right\rfloor - \left\lfloor\frac{r^2}{4i+3}\right\rfloor\right)$$

The second solution has a much nicer form if the sum of squares is defined as the number of ways of writing n as the sum of two squares:

Solution 2.

$$N(r) = \sum_{n=0}^{r^2} r_2(n)$$

17 Appendix B: Base Security Objectives

There are four basic security objectives that must be considered when constructing any system concerned with securing data or information. These four basic definitions allow for the derivation of all other security objectives which may or may not be necessary to ensure the security of information for a given system.

- **Confidentiality:** The objective of confidentiality ensures that unauthorized users will not be purposefully (or accidentally) give access to resources protected by the system.
- **Integrity:** Ensures that the resources are preserved, used, and appropriately maintained throughout their life-cycle under the system. That is, any data is not alterable in an undetectable manner, retains the same accuracy as its created date (or registered modification date), and is complete with respect to its creation and activity log.
- **Availability:** Ensuring resources are available to authorized parties as required.
- **Non-repudiation:** Prevention of commitment or action denial.

17.1 Derived Security Objectives

Each implementation of a scheme requires a flexible set of security objectives which are dependent upon the context the system and it's users will employ **TODO:change wording**. The base definitions for the objectives allow the derivation of all other security objectives.

Confidentiality	Keeping information in the hands of only authorized or trusted users.
Data Integrity	Ensuring information remains unaltered by unauthorized parties.
Identification	Corroborating the identity of a user, system, account, etc.
Message Authentication	Corroborating the source of information.
Signature	An object which assigns information to a user.
Authorization	Permissions given to another user to allow them to use or be something.
Validation	An object which ensures that authorization is provided in an appropriate time frame.
Access Control	Restricting the access of resources to only authorized users.
Certification	Information endorsed by an authorized party.
Timestamping	A record of the creation. existence, or modification of information.
Witnessing	A verification of the creation, existence, or modification of information by a user who did not take part in the activity.
Receipt	Confirmation that information has been received.
Confirmation	Affirmation of a provided service.
Ownership	Provision of usage or transfer rights to a user.
Anonymity	A process which conceals the identity of a user.
Non-repudiation	Prevention of commitment or action denial.
Revocation	Allows the privileges of an authorized user to be reversed.

Table 8: Derivable Security Objectives

A3: Replace Me

18 Appendix D: Time Complexity

Time complexity quantifies the amount of time an algorithm requires to solve a problem as a function of the length of the input it is provided.

The time complexity is typically measured by counting the number of operations performed, where the amount of time taken by each operation is fixed, and the number of operations is known. The time complexity can then be represented asymptotically by allowing the input n to approach infinity.

However, algorithms may not be well behaved for all types of input, even when they are the same size. That is, the performance (runtime), T of an algorithm A, may vary greatly for different types or sizes of inputs n. An algorithm is classified according to its runtime behavior for an arbitrary input size.

These behaviors are typically given in terms of the *worst-case* time complexity of an algorithm, $T(n)$.
Definition 35. *Worst-case Time Complexity*

The worst-case time complexity of an algorithm is the largest possible amount of time a given algorithm may require in order to solve a problem.

Insert notes.

19 Appendix E: Lattices Laws

A lattice is a partially ordered set (poset) such that for

Let Λ be a lattice of $\mathbb{R}^n$, and let a, b, and c be elements of Λ. Then Λ satisfies the following laws (axioms).
Definition 36. *Identity Laws (If Bounded Lattice)*

$$\begin{aligned} a \vee 0 &= a \\ a \wedge 1 &= a \end{aligned}$$

Definition 37. *Commutative Laws*

$$\begin{aligned} a \vee b &= b \vee a \\ a \wedge b &= b \wedge a \end{aligned}$$

Definition 38. *Associative Laws*

$$\begin{aligned} a \vee (b \vee c) &= (a \vee b) \vee c \\ a \wedge (b \wedge c) &= (a \wedge b) \wedge c \end{aligned}$$

Definition 39. *Absorbtion Laws*

$$\begin{aligned} a \vee (a \wedge b) &= a \\ a \wedge (a \vee b) &= a \end{aligned}$$

Definition 40. *Idempotent Laws*

$$a \vee a = a,$$
$$a \wedge a = a$$

Appendix E: Morphisms of Lattices

Definition 41. *Morphism of Lattices A morphism of lattices* Λ *and* Δ *is a function* $f : \Lambda \to \Delta$ *such that for all elements* $a, b \in \Lambda$ *we have the following:*

$$f(a \vee_\Lambda b) = f(a) \vee_\Delta f(b)$$
$$f(a \wedge_\Lambda b) = f(a) \wedge_\Delta f(b)$$

If Λ and Δ are bounded lattices then the following properties we need the following two properties in addition to the two given above:
Definition 42. *Additional Properties for Morphisms of Bounded Lattices*

$$f(0_\Lambda) = f(0_\Delta)$$
$$f(1_\Lambda) = f(1_\Delta)$$

Appendix F: NC (Nik's Class) Complexity

NC (Nick's Class) is the set of decision problems parallelized over a polynomial number of processors which are decidable in polylog time. That is, the set of problems which are efficiently solved by parallel computers.

An equivalent (and at a general level somewhat more relevant) definition to the parallel processor definition of NC, is for the class defined in terms of Boolean circuits.

For the Boolean statement, NC is the set of decision problems that can be calculated from the length of the given input (solved by uniform Boolean circuits) over a polynomial number of logic gates with at most two inputs, where the circuit depth is polylog $\mathcal{O}(\log^i n$.

20 Appendix G: Basic Definitions for Rings

A ring $\mathcal{R}$ is a group defined for two operations, instead of only one operator. A ring has both additive operation, and a multiplicative operations.

Denoted:

$\mathcal{R}(+, *)$.

Let $\mathbb{F}$ be a field, then $\mathbb{F}$ is a ring where the nonzero elements of $\mathbb{F}$ form an abelian group under multiplication. We can do anything in this field we can in the reals, complex, or rational numbers we are familiar with; we can add, multiply, subtract, and divide. In fact the set of reals, complex, and rational numbers are all fields themselves.

A multiplicative group is is the set of all invertible elements of a field, also a ring, or some other structure having a multiplicative operation on it's elements. For example, $\mathbb{Z}/n\mathbb{Z}$ is the multiplicative group of integers mod n; i.e., the elements of $\mathbb{Z}/n\mathbb{Z}$ which are invertible under multiplication.
Note 13. Fields $\subset$ groups, Rings $\subset$ groups, Fields $\subset$ rings,

Notice: While all rings are groups (abelian groups under addition), the converse is not always true, all rings are not fields.

A ring is a set $\mathcal{R}$ defined on two operations $+$ and $*$, which satisfy the following properties:

- The additive identity 0 exists,
- the elements of $\mathcal{R}$ commute over addition,
- they also are associative over addition,
- an additive inverse exists for every element in $\mathcal{R}$.

These four properties define an abelian group. The next three properties are defined similarly, with an exception of the multiplicative inverse.

- A multiplicative identity exists, 1.
- the elements of $\mathcal{R}$ commute over multiplication,
- they also are associative over multiplication.

- The distributive law, holds. (It's the glue that connects these two operations together).

19

[19] The distributive property: the 'jelly side down' of ring theory.

Todo list

References

[1] *Paillier's cryptosystem revisited*, ACM, New York, New York, USA, nov 2001.

[2] *An Elementary Proof of the Quantum Adiabatic Theorem*, arXiv.org (2004).

[3] *Introduction to post-quantum cryptography*, Post-quantum cryptography, Springer, Berlin, Berlin, Heidelberg, 2009, pp. 1–14.

[4] *Adaptive trapdoor functions and chosen-ciphertext security*, Advances in Cryptology – EUROCRYPT 2010, Springer, Berlin, 2010, pp. 673–692.

[5] *Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes*, Public Key Cryptography – PKC 2010, Springer Berlin Heidelberg, Berlin, Heidelberg, may 2010, pp. 420–443.

[6] *The LLL Algorithm*, Information Security and (2010).

[7] *Some mathematical mysteries in lattices*, (2012).

[8] *Classical hardness of learning with errors*, arXiv.org (2013).

[9] *An introduction to mathematical cryptography*, second ed., Undergraduate Texts in Mathematics, Springer, New York, New York, NY, 2014.

[10] Martin Abadi and Phillip Rogaway, *Reconciling two views of cryptography (the computational soundness of formal encryption)*, J. Cryptol. **20** (2007), no. 3, 395–395.

[11] M. Ajtai, *Generating hard instances of lattice problems (extended abstract)*, Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '96, ACM, 1996, pp. 99–108.

[12] Mikl6s Ajtai, Janos Komls, and Endre Szemerdi, *Generating expanders from two permutations*, A tribute to Paul Erds, edited by A. Baker, B. Bollobs & A. Hajnal (1990), 1–12.

[13] Miklos Ajtai, *Worst-case complexity, average-case complexity and lattice problems*, (1998).

[14] Miklós Ajtai and Cynthia Dwork, *A public-key cryptosystem with worst-case/average-case equivalence*, Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '97, ACM, 1997, pp. 284–293.

[15] Benny Applebaum, Boaz Barak, and Avi Wigderson, *Public-key cryptography from different assumptions*, Proceedings of the forty-second ACM symposium on Theory of computing, ACM, 2010, pp. 171–180.

[16] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai, *Fast cryptographic primitives and circular-secure encryption based on hard learning problems*, CRYPTO '09: Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology (Berlin, Heidelberg), Springer-Verlag, 2009, pp. 595–618.

[17] Frederik Armknecht, Stefan Katzenbeisser, and Andreas Peter, *Group homomorphic encryption: Characterizations, impossibility results, and applications*, Des. Codes Cryptography **67** (2013), no. 2, 209–232.

[18] Christian Badertscher, Christian Matt, Ueli Maurer, Phillip Rogaway, and Bjrn Tackmann, *Robust authenticated encryption and the limits of symmetric cryptography.*

[19] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway, *Relations among notions of security for public-key encryption schemes*, Advances in CryptologyCRYPTO'98, Springer, 1998, pp. 26–45.

[20] Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, and Till Stegers, *Selected areas in cryptography*, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 295–312.

[21] Mihir Bellare and Phillip Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, Proceedings of the 1st ACM Conference on Computer and Communications Security (New York, NY, USA), CCS '93, ACM, 1993, pp. 62–73.

[22] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani, *Strengths and weaknesses of quantum computing*, SIAM J. Comput. **26** (1997), no. 5, 1510–1523.

[23] Jean-Christophe Benoist, *algorithme de grover*.

[24] Jean-François Biasse and Fang Song, *On the quantum attacks against schemes relying on the hardness of finding a short generator of an ideal in q (ζpn)*, (2015).

[25] Dan Boneh and Ramarathnam Venkatesan, *Breaking rsa may not be equivalent to factoring*, Advances in CryptologyEUROCRYPT'98, Springer, 1998, pp. 59–71.

[26] Daniel R. L. Brown, *Breaking rsa may be as difficult as factoring*, Cryptology ePrint Archive, Report 2005/380, 2005.

[27] Jin-Yi Cai and Ajay P Nerurkar, *An improved worst-case to average-case connection for lattice problems*, focs, IEEE, 1997, p. 468.

[28] Peter Campbell, Michael Groves, and Dan Shepherd, *Soliloquy: A cautionary tale*, ETSI 2nd Quantum-Safe Crypto Workshop, 2014.

[29] Ronald Cramer, Lo Ducas, Chris Peikert, and Oded Regev, *Recovering short generators of principal ideals in cyclotomic rings*, 2015.

[30] Angsuman Das, Sabyasachi Dutta, and Avishek Adhikari, *Indistinguishability against chosen ciphertext verification attack revisited: The complete picture*, Provable Security, Springer, 2013, pp. 104–120.

[31] Alexander W Dent, *Fundamental problems in provable security and cryptography*, Philosophical Transactions: Mathematical, Physical and Engineering Sciences **364** (2006), no. 1849, 3215–3230.

[32] Stephan Ramon Garcia, Trevor Hyde, and Bob Lutz, *Gauss's Hidden Menagerie: From Cyclotomy to Supercharacters*, Notices of the American Mathematical Society **62** (2016), no. 01, 7–21.

[33] Craig Gentry, *Fully homomorphic encryption using ideal lattices*, Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '09, ACM, 2009, pp. 169–178.

[34] Oded Goldreich, Shafi Goldwasser, and Shai Halevi, *Collision-free hashing from lattice problems*, Electronic Colloquium on Computational Complexity (ECCC), vol. 3, 1996, pp. 236–241.

[35] Shafi Goldwasser and Silvio Micali, *Probabilistic encryption / how to play mental poker keeping secret all partial information*, Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '82, ACM, 1982, pp. 365–377.

[36] Matthew Green, *Poker is hard especially for cryptographers*, 2011.

[37] ———, *The many flaws of dual ec drbg*, Sep 2013.

[38] ———, *Hopefully the last post ill ever write on dual ec drbg*, Jan 2015.

[39] Brett Hemenway and Rafail Ostrovsky, *On homomorphic encryption and chosen-ciphertext security*, Public Key Cryptography–PKC 2012, Springer, 2012, pp. 52–65.

[40] Yali Jiang and Xiuling Ju, *Lattice-Based CCA-Secure Cryptosystem from IBE System*, vol. 2, IEEE, 2009.

[41] Daniel Kroening and Ofer Strichman, *Decision procedures: an algorithmic point of view*, Springer Science & Business Media, 2008.

[42] V Satish Kumar and V Trilik Kumar, *Securing data using fully homomorphic encryption schemes over cloud computing*, Indexing Journal Indexing: Our journal has recently joined International Database for Indexing with DOAJ, Index Copernicus, Open J Gate, CAS, Google Scholar, WAME... More≫ (2014).

[43] Arjen K. Lenstra, *Factorization of polynomials*, SIGSAM Bull. **18** (1984), no. 2, 16–18.

[44] Vadim Lyubashevsky, *On Ideal Lattices and Learning with Errors over Rings*, Journal of the Acm **60** (2013), no. 6.

[45] Vadim Lyubashevsky, Chris Peikert, and Oded Regev, *On ideal lattices and learning with errors over rings*, Proceedings of the 29th Annual International Conference on Theory and Appli-

cations of Cryptographic Techniques (Berlin, Heidelberg), EUROCRYPT'10, Springer-Verlag, 2010, pp. 1–23.

[46] Shai Halevi Vinod Vaikuntanathan Marten van Dijk, Craig Gentry, *Fully Homomorphic Encryption over the Integers*, Advances in Cryptology EUROCRYPT 2010 (Eurocrypt2010, ed.), 6110, vol. 2010, Springer Berlin Heidelberg, Berlin, Heidelberg, may 2010, pp. 24–43.

[47] Daniele Micciancio, *Improved cryptographic hash functions with worst-case/average-case connection*, Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, ACM, 2002, pp. 609–618.

[48] Daniele Micciancio and Shafi Goldwasser, *Complexity of lattice problems*, 1st ed., A Cryptographic Perspective, Springer US, Boston, MA, 2002.

[49] Daniele Micciancio and Chris Peikert, *Trapdoors for lattices: simpler, tighter, faster, smaller*, EUROCRYPT'12: Proceedings of the 31st Annual international conference on Theory and Applications of Cryptographic Techniques (Berlin, Heidelberg), Springer-Verlag, 2012, pp. 700–718.

[50] Daniele Micciancio and Oded Regev, *Worst-case to average-case reductions based on gaussian measures*, SIAM Journal on Computing **37** (2007), no. 1, 267–302.

[51] Toms Oliveira e Silva, Siegfried Herzog, and Silvio Pardi, *Empirical verification of the even goldbach conjecture and computation of prime gaps up to 4 10^1*, Mathematics of Computation **83** (2014), no. 288, 2033–2060.

[52] Chris Peikert, *Some recent progress in lattice-based cryptography*, TCC '09: Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography (Berlin, Heidelberg), Springer-Verlag, 2009, pp. 72–72.

[53] ______, *Heuristics and rigor in lattice-based cryptography*, SCN'10: Proceedings of the 7th international conference on Security and cryptography for networks (Berlin, Heidelberg), Springer-Verlag, 2010, pp. 54–54.

[54] ______, *Lattice cryptography for the internet*, Post-Quantum Cryptography, Springer, 2014, pp. 197–219.

[55] ______, *A decade of lattice cryptography*, 2015.

[56] Chris Peikert and Alon Rosen, *Lattices that admit logarithmic worst-case to average-case connection factors*, STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing (New York, NY, USA), ACM, 2007, pp. 478–487.

[57] Chris Peikert and Vinod Vaikuntanathan, *Noninteractive statistical zero-knowledge proofs for lattice problems*, CRYPTO 2008: Proceedings of the 28th Annual conference on Cryptology (Berlin, Heidelberg), Springer-Verlag, 2008, pp. 536–553.

[58] Chris Peikert and Brent Waters, *Lossy trapdoor functions and their applications*, SIAM J. Comput. **40** (2011), no. 6, 1803–1844.

[59] Christophe Petit, Jean-Jacques Quisquater, et al., *Rubik's for cryptographers.*, IACR Cryptology ePrint Archive **2011** (2011), 638.

[60] M. O. Rabin, *Digitalized signatures and public-key functions as intractable as factorization*, Tech. report, Cambridge, MA, USA, 1979.

[61] Oded Regev, *New lattice-based cryptographic constructions*, J. ACM **51** (2004), no. 6, 899–942.

[62] ______, *On lattices, learning with errors, random linear codes, and cryptography*, Journal of the Acm **56** (2009), no. 6, 34–40.

[63] Thomas Ristenpart and Phillip Rogaway, *How to enrich the message space of a cipher*, Proceedings of the 14th International Conference on Fast Software Encryption (Berlin, Heidelberg), FSE'07, Springer-Verlag, 2007, pp. 101–118.

[64] Ronald L Rivest, Adi Shamir, and Len Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), no. 2, 120–126.

[65] P Rogaway, *On the role definitions in and beyond cryptography*, Advances in Computer Science - Asian 2004, Proceedings **3321** (2004), no. 6, 13–32.

[66] Phillip Rogaway, *The moral character of cryptographic work*, (2015).

[67] Phillip Rogaway, Mark Wooding, and Haibin Zhang, *The security of ciphertext stealing*, Proceedings of the 19th International Conference on Fast Software Encryption (Berlin, Heidelberg), FSE'12, Springer-Verlag, 2012, pp. 180–195.

[68] Gian-Carlo Rota, *The many lives of lattice theory*, Notices of the American Mathematical Society **44** (1997), no. 11, 1440–1445.

[69] Markus Rckert and Michael Schneider, *Estimating the security of lattice-based cryptosystems.*, IACR Cryptology ePrint Archive **2010** (2010), 137.

[70] Bruce Schneier, *Cryptographic design vulnerabilities*, Computer **31** (1998), no. 9, 29–33.

[71] ______, *The strange story of dual ec drbg*, Nov 2007.

[72] Alice Silverberg, *Fully homomorphic encryption for mathematicians*, Women in Numbers 2: Research Directions in Number Theory **606** (2013), 111.

[73] Joseph H Silverman, *An introduction to the theory of lattices and applications to cryptography*, Computational Number Theory and Applications to Cryptography, University of Wyoming, Jun (2006).

[74] Fang Song, *A note on quantum security for post-quantum cryptography*, IACR Cryptology **8772** (2014), no. Chapter 15, 246–265.

[75] Sophos, *Threatsaurus - the a-z list of data security threats for computers, networks and endusers*, 2015.

[76] Ryan Stanley, *Soliloquy cryptographic primitive*, 2010.

[77] John Steinberger, *Stam's collision resistance conjecture*, Advances in Cryptology EUROCRYPT 2010, Springer Berlin Heidelberg, Berlin, Heidelberg, may 2010, pp. 597–615.

[78] Patrick Struck, *On the difference between hardness and security: a comparison of lattice-based signature schemes*, Bachelor thesis, TU Darmstadt, June 2015.

[79] Paul Syverson, *Why i'm not an entropist*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[80] T. Vidick, *Three-player entangled xor games are np-hard to approximate*, IEEE (2013), 766–775.

[81] Kathleen Ward and Phillip Rogaway, *When to hyphenate phrases such as 'Public Key'*, Tech. report, University of California, Davis, December 2015.