Chapter 1 Introduction

Cryptography is the science of keeping private information from unauthorized access, of ensuring data integrity and authentication, and other tasks[1]. In this seminar, we will focus on post quantum cryptography and in particular will discuss their security. Before going to Post quantum cryptography, a brief review of classical cryptography and its current challenges are discussed here. Two parties, Alice(Sender) and Bob(Recipient), wish to exchange messages via some insecure channel in a way that protects their messages from eavesdropping. An algorithm, which is called a cipher in this context, scrambles(encryption) Alice's message via some rule such that restoring the original message is hard-if not impossible-without knowledge of the secret key. This "scrambled" message is called the Ciphertext. On the other hand, Bob (who possesses the secret key) can easily decipher Alice's ciphertext and obtains her original plaintext. Figure 1 in this section presents the conventional cryptosystem.

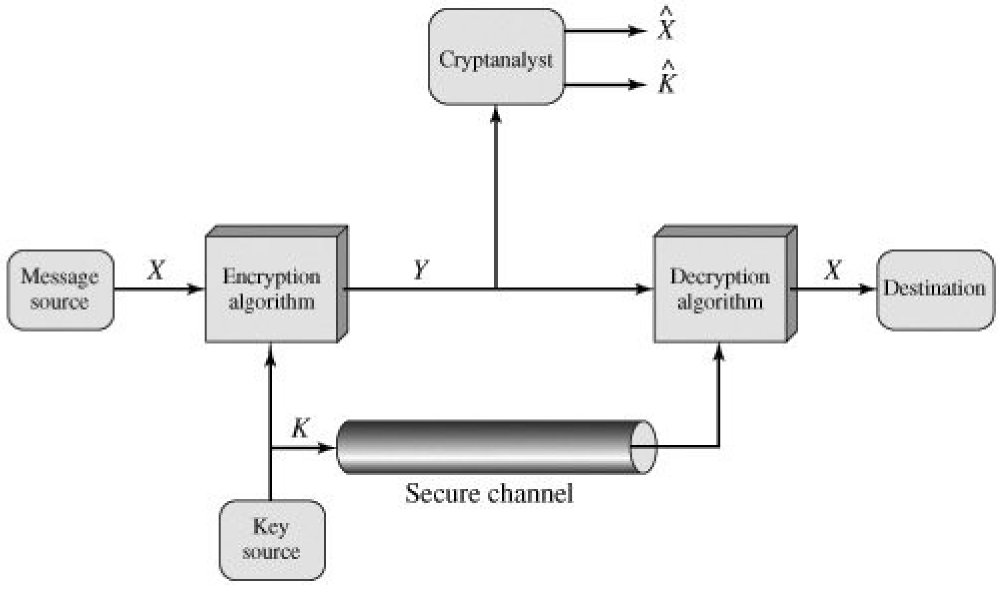

Fig 1 Conventional Cryptosystem[1]

The purpose of cryptography is to transmit information such that only the intended recipient is able to view it. Although the field of cryptography is ancient, it is not static. Cryptographic techniques have evolved over the centuries, with the code-makers working to stay ahead of the code-breakers. The next major step in this evolutionary process may be at hand. Today’s most common encryption methods are threatened by the potential creation of the quantum computer. But already quantum cryptography has been developed which promises more secure communication than any existing technique and cannot be compromised by quantum computers. Quantum cryptography takes advantage of the unique and unusual behaviour of microscopic objects to enable users to securely develop secret keys as well as to detect eavesdropping.

Although work on quantum cryptography was begun by Stephen J. Wiesner in the late 1960’s, the first protocol for sending a private key using quantum techniques was not published until 1984 by Bennett and Brassard[1]. The development of quantum cryptography was motivated by the short-comings of classical cryptographic methods, which can be classified as either “public-key” or “secret-key” methods. Public-key encryption is based on the idea of a safe with two keys: a public key to lock the safe and a private key to open it. Using this method, anyone can send a message since the public key is used to encrypt messages, but only someone with the private key can decrypt the messages. Since the encrypting and decrypting keys are different, it is not necessary to securely distribute a key. The security of public-key encryption depends on the assumed difficulty of certain mathematical operations, such as factoring extremely large prime numbers. There are two problems with basing security on the assumed difficulty of mathematical problems. The first problem is that the difficulty of the mathematical problems is assumed, not proven. All security will vanish if efficient factoring algorithms are discovered. The second problem is the threat of quantum computers. The theoretical ability of quantum computers to essentially process large amounts of information in parallel would remove the time barrier to factoring large numbers. Thus, public-key encryption, though secure at the moment, faces a serious threat as quantum computing comes closer to reality.

Secret-key encryption requires that two users first develop and securely share a secret key, which is a long string of randomly-chosen bits. The users then use the secret key along with public algorithms to encrypt and decrypt messages. There are two main problems with secret-key encryption. The first problem is that by analysing the publicly-known encrypting algorithm, it sometimes becomes easier to decrypt the message. This problem can be somewhat offset by increasing the length of the key. The second problem is securely distributing the secret key in the first place. This is the well- known “key-distribution problem”.

Post Quantum cryptography solves the problems of secret-key cryptography by providing a way for two users who are in different locations to securely establish a secret key and to detect if eavesdropping has occurred.

Chapter 2 Post Quantum Cryptography

Post-quantum cryptography refers to research on cryptographic primitives (usually public-key cryptosystems) that are not breakable using quantum computers. This term came about because most currently popular public-key cryptosystems rely on the integer factorization problem or discrete logarithm problem, both of which would be easily solvable on large enough quantum computers using Shor's algorithm[2]. Even though current publicly known experimental quantum computing is nowhere near powerful enough to attack real cryptosystems, many cryptographers are researching new algorithms, in case quantum computing becomes a threat in the future. This work is popularized by the PQCrypto conference series since 2006. . In contrast, most current symmetric cryptography (symmetric ciphers and hash functions) is secure from quantum computers. The quantum Grover's algorithm can speed up attacks against symmetric ciphers, but this can be counteracted by increasing key size. Thus post-quantum cryptography does not focus on symmetric algorithms

2.1 Need of Post quantum Cryptography

In a predictive sense, quantum computers may become a technological reality; it is therefore important to study cryptographic schemes that are (supposedly) secure even against adversaries with access to a quantum computer. The study of such schemes is often referred to as post-quantum cryptography. The need for post-quantum cryptography arises from the fact that many popular encryption and signature schemes (such as RSA and its variants, and schemes based on elliptic curves) can be broken using Shor's algorithm[2] for factoring and computing discrete logarithms on a quantum computer. Examples for schemes that are (as of today's knowledge) secure against quantum adversaries are McEliece and lattice-based schemes.

2.2 Approaches of Post Quantum Cryptography

Currently post-quantum cryptography is mostly focused on four different approaches:

- **Lattice-based cryptography** such as NTRU and GGH[4]
- **Multivariate cryptography** such as Unbalanced Oil and Vinegar[3]
- **Hash-based signatures** such as Lamport signatures and Merkle signature scheme
- **Code-based cryptography** that relies on error-correcting codes, such as McEliece encryption and Niederreiter signatures.

However this report mainly focuses on Lattice based and Hash based Cryptography

2.2.1 Lattice based Cryptography

Lattice-based cryptography is the generic term for asymmetric cryptographic primitives based on lattices[4]. Lattices were first studied by mathematicians Joseph Louis Lagrange and Carl Friedrich Gauss. Lattices have been used recently in computer algorithms and in cryptanalysis. A lattice L is a set of points in the n-dimensional Euclidean space $\mathbf{R}^n$ with a strong periodicity property. A basis of L is a set of vectors such that any element of L is uniquely represented as their linear combination with integer coefficients. When n is at least 2, each lattice has infinitely many different bases.

Mathematical problems are used to construct a cryptography system. These problems are usually hard to solve unless one has extra information. Mathematical problems based on lattices are the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP)[5]. *SVP*: Given a basis of a lattice, find the shortest vector in the lattice. *CVP*: Given a basis of a lattice and a vector not in the lattice, find the lattice vector with the least distance to the first vector. These problems are normally hard to solve. There are algorithms to solve these problems with a good basis.

Lattice basis reduction is a transformation of an integer lattice basis in order to find a basis with short, nearly orthogonal vectors. If one can compute such a lattice basis, the CVP and SVP problems are easy to solve.

2.2.2 Multivariate based Cryptography

Multivariate cryptography is the generic term for asymmetric cryptographic primitives based on multivariate polynomials over finite fields [3]. In certain cases those polynomials could be defined over both a ground and an extension field. If the polynomials have the degree two, we talk about multivariate quadratics. Solving systems of multivariate polynomial equations is proven to be NP-Hard or NP-Complete[3]. That's why those schemes are often considered to be good candidates for post-quantum cryptography, once quantum computers can break the current schemes. Today multivariate quadratics could be used only to build signatures. All attempts to build a secure encryption scheme have so far failed.

2.2.3 Hash based signature scheme

Hash-based digital signature schemes use a cryptographic hash function. Their security relies on the collision resistance of that hash function. In fact, hash-based digital signature schemes that are secure if and only if the underlying hash function is collision resistant. The existence of collision resistant hash functions can be viewed as a minimum requirement for the existence of a digital signature scheme that can sign many documents with one private key.
That signature scheme maps documents (arbitrarily long bit strings) to digital signatures (bit strings of fixed length). This shows that digital signature algorithms are in fact hash functions. Those hash functions must be collision resistant: if it were possible to construct two documents with the same digital signature, the signature scheme could no longer be considered secure. This argument shows that there exist hash-based digital signature schemes as long as there exists any digital signature scheme that can sign multiple documents
using one private key. As a consequence, hash-based signature schemes are the most important post-quantum signature candidates. Although there is no proof of their quantum computer resistance, their security requirements are minimal. Also, each new cryptographic hash function yields a new hash-based signature scheme. So the construction of secure signature schemes is independent of hard algorithmic problems in number theory or algebra.

2.2.4 Code Based Cryptography

It is the cryptosystems in which the algorithmic primitive (the underlying one-way function) uses an error correcting code C. This primitive may consist in adding an error to a word of C or in computing a syndrome relatively to a parity check matrix of C.
The first of those systems is a public key encryption scheme and it was proposed by Robert J. McEliece in 1978 [48]. The private key is a random binary irreducible Goppa code and the public key is a random generator matrix of a randomly permuted version of that code. The ciphertext is a codeword to which some errors have been added, and only the owner of the private key (the Goppa code) can remove those errors. Three decades later, some parameter

adjustment have been required, but no attack is known to represent a serious threat on the system, even on a quantum computer.

2.3 Hash Based Signature Scheme

Hash-based signature schemes were invented by Ralph Merkle [7]. Merkle started from one-time signature schemes, in particular that of Lamport and Diffie [6]. One-time signatures are even more fundamental. The construction of a secure one-time signature scheme only requires a one-way function, which are necessary and sufficient for secure digital signatures. So one-time signature schemes are the most fundamental type of digital signature schemes. However, they have a severe disadvantage. One key-pair consisting of a secret signature key and a public verification key can only be used to sign and verify a single document. This is inadequate for most applications. It was the idea of Merkle to use a hash tree that reduces the validity of many one-time verification keys (the leaves of the hash tree) to the validity of one public key (the root of the hash tree).

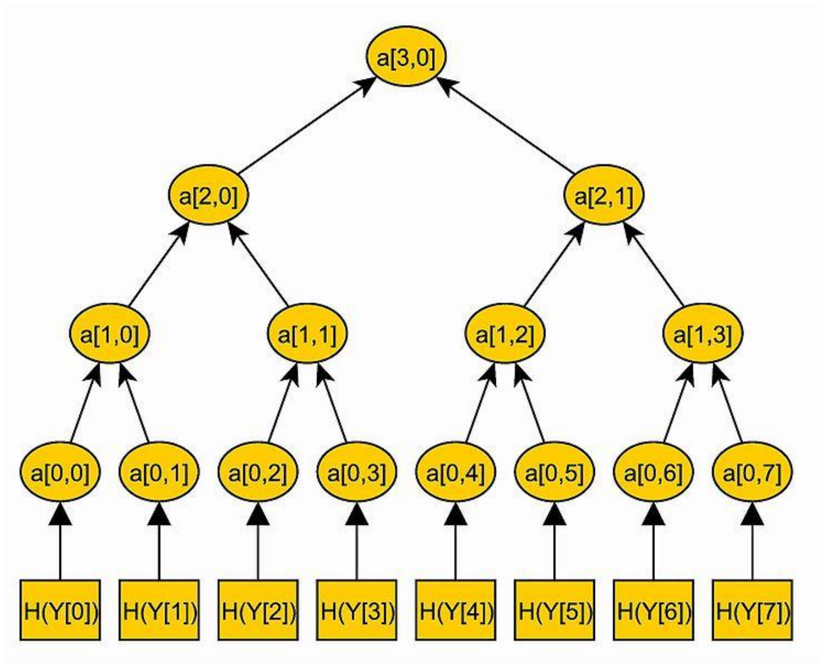

Fig 2 Merkle Hash tree[6]

Merkle signature scheme

Key Generation

The Merkle Signature Scheme can only be used to sign a limited number of messages with one public key pub. The number of possible messages must be a power of two, so that we denote the possible number of messages as $N=2^n$.

The first step of generating the public key pub is to generate the public keys X_i and private keys Y_i of 2^n one-time signatures. For each public key X_i, with $1 \leq i \leq 2^n$, a hash value $h_i = H(X_i)$ is computed. With these hash values h_i a hash tree is built.

We call a node of the tree $a_{i,j}$, where i denote the level of the node. The level of a node is defined by the distance from the node to a leaf. Hence, a leaf of the tree has level $i = 0$ and the root has level $i = n$. We number all nodes of one level from the left to the right, so that $a_{i,0}$ is the leftmost node of level i.

In the Merkle Tree the hash values h_i are the leaves of a binary tree, so that $h = a_{0,i}$. Each inner node of the tree is the hash value of the concatenation of its two children.

So $a_{1,0} = H(a_{1,0} \| a_{1,1})$ and $a_{2,0} = H (a_{1,0} \| a_{1,1})$. (1)

In this way, a tree with 2^n leaves and $2^{n+1} - 1$ nodes is built. The root of the tree $a_{n,0}$ is the public key pub of the Merkle Signature Scheme.

Signature Generation

To sign a message M with the Merkle Signature Scheme, the message M is signed with a one-time signature scheme, resulting in a signature sig', first. This is done, by using one of the public and private key pairs (X_i , Y_i).

The corresponding leaf of the hash tree to a one-time public key X_i is $a_{0,i} = H(X_i)$. We call the path in the hash tree from $a_{0,i}$ to the root A. The path A consists of n + 1 nodes, $A_0,\ldots,A_n$, with $A_0 = a_{0,i}$ being the leaf and $A_n = a_{n,0}$ = pub being the root of the tree. To compute this path A, we need every child of the nodes $A_1,\ldots,A_n$. We know that A_i is a child of A_{i+1}. To calculate the next node A_{i+1} of the path A, we need to know both children of A_{i+1} . So we need the brother node of A_i. We call this node $auth_i$, so that

$Ai + 1 = H\ (\ Ai\ ||auth_i\ )$. Hence, n nodes $auth_0, \ldots, auth_{n-1}$ are needed, to compute every node of the path A. We now calculate and save these nodes $auth_0, \ldots, auth_{n-1}$.

These nodes, plus the one-time signature sig' of M is the signature

$$sig = (sig'\ ||auth0\ ||auth1\ ||\ldots||authn - 1) \tag{2}$$

of the Merkle Signature Scheme. An example of an authentication path is illustrated in the figure on the right.

Signature Verification

The receiver knows the public key pub, the message M, and the signature

$$sig = (sig'\ ||auth_0\ ||auth_1\ ||\ldots||auth_{n-1}) \tag{3}$$

At first, the receiver verifies the one-time signature sig' of the message M, If sig' is a valid signature of M, the receiver computes $A0 = H(Xi)$ by hashing the public key of the one-time signature. For j=1,…..,n-1, the nodes of A_j of the path A are computed with $Aj = H(\ Aj - 1\ ||\ authj - 1\ )$. If A_n equals the public key pub of the Merkle signature scheme, the signature is valid.

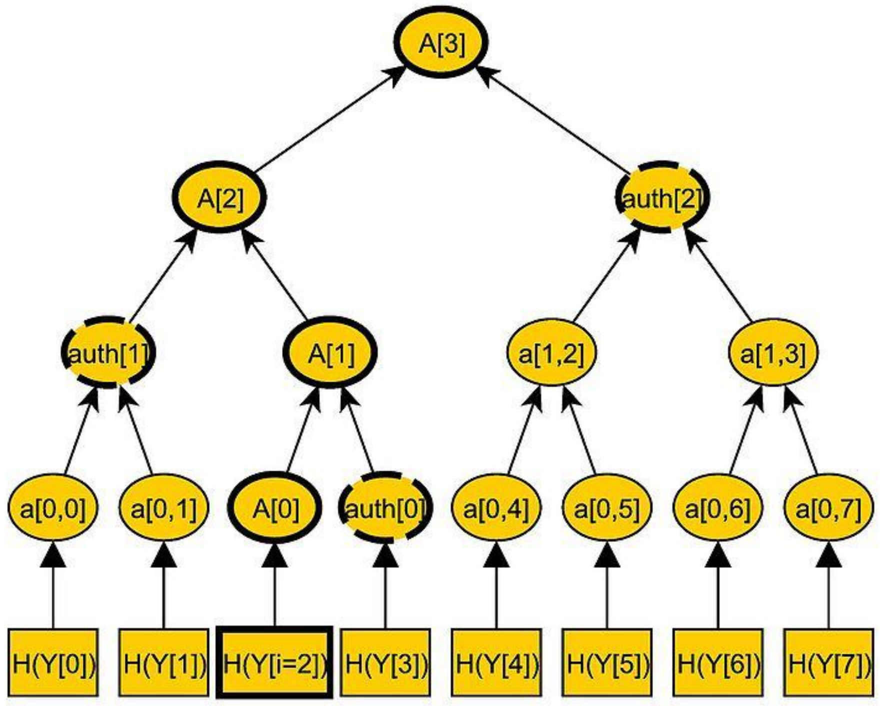

Fig 3 Merkle Tree with path A and authentication path for i =2[6]

2.4 Lattice based Cryptography

Lattice-based cryptographic constructions hold a great promise for post-quantum cryptography, as they enjoy very strong security proofs based on worst-case hardness, relatively efficient implementations, as well as great simplicity[8]. In addition, lattice-based cryptography is believed to be secure against quantum computers. A lattice is a set of points in n-dimensional space with a periodic structure, More formally, given n-linearly independent vectors $b_1, \ldots, b_n \in R_n$, the lattice generated by them is the set of vectors

$$L(b1, \quad , \quad , \quad , \quad bn) = \sum xibi\,; xi \ \in Z$$

The vectors $b_1, \ldots, b_n$ are known as a basis of the lattice.

The way lattices can be used in cryptography is by no means obvious, and was discovered in a breakthrough paper by Ajtai [7]. His result has by now developed into a whole area of research whose main focus is on expanding the scope of lattice-based cryptography and on creating more practical lattice based cryptosystems.

Lattice based has a famous cryptosystem called NTRUEncrypt public key cryptosystem[8]. NTRUEncrypt public key cryptosystem is a lattice-based alternative to RSA and ECC and is based on the shortest vector problem in a lattice (i.e. is not known to be breakable using quantum computers)[8]. Operations are based on objects in a truncated polynomial ring with convolution multiplication and all polynomials in the ring as integer coefficients and degree at most N-1:

$$a = a_0 + a_1X + a_2X^2 + \ldots\ldots\ldots\ldots + a_{n-1} X^{n-1} \qquad (4)$$

NTRU is actually a parameterised family of cryptosystems; each system is specified by three integer parameters (N, p, q) which represent the maximal degree N-1 for all polynomials in the truncated ring R, a small modulus and a large modulus, respectively, where it is assumed that N is prime, q is always larger than p, and p and q are coprime; and four sets of polynomials L_f, L_g, L_m and L_r (a polynomial part of the private key, a polynomial for generation of the public key, the message and a blinding value, respectively), all of degree at most N-1

Key Generation

Sending a secret message from Alice to Bob requires the generation of a public and a private key. The public key is known by both Alice and Bob and the private key is only known by Bob. To generate the key pair two polynomials **f** and **g**, with coefficients much smaller than q, with degree at most N-1 and with coefficients in {-1, 0, 1} are required. They can be considered as representations of the residue classes of polynomials modulo X^N - 1 in R. The polynomial $f \in Lf$ must satisfy the additional requirement that the inverses modulo q and modulo p (computed using the Euclidean algorithm) exist, which means that

$f\,.fp = 1\ (mod\ p)$ and $f\,.fq = 1\ (mod\ p)$ must hold. So when the chosen f is not invertible, Bob has to go back and try another f.

Both f and f_p is Bob’s private key. The public key **h** is generated computing the quantity

$$h = fq.g\ (mod\ q) \qquad (5)$$

Encryption

Alice, who wants to send a secret message to Bob, puts her message in the form of a polynomial m with coefficients {-1, 0, 1}. In modern applications of the encryption, the message polynomial can be translated in a binary or ternary representation. After creating the message polynomial, Alice chooses randomly a polynomial **r** with small coefficients (not restricted to the set {-1, 0, 1}), that is meant to obscure the message.

With Bob's public key **h** the encrypted message **e** is computed:

$$e = pr.h + m\ (mod\ q) \tag{6}$$

This ciphertext hides Alice's messages and can be sent safely to Bob.

Decryption

The ciphertext can be decrypted using the following formula

$$a = e\ (mod\ q) \tag{7}$$

By rewriting the polynomials, this equation is actually representing the following computation:

$$a = f.e\ (mod\ q) \tag{8}$$

$$a = f.(r.ph + m)(mod\ q) \tag{9}$$

$$a = f.(r.p(fq).\ g + m)(mod\ q) \tag{10}$$

$$a = p(r.g) + f.m\ (mod\ q) \tag{11}$$

The next step will be to calculate a mod p:

$$b = a\ (mod\ p) \tag{12}$$

$$b = f.m\ (mod\ p) \text{ (because } \ pr.g\ (mod\ p) = 0) \tag{13}$$

Knowing **b** Bob can use the other part of his private key (f_p) to recover Alice's message by multiplication of **b** and f_p

$$c = fp.b = fp.f.m\ (mod\ p) \tag{14}$$

$$c = m\ (mod\ p) \tag{15}$$

since the property $f.fp = 1\ (mod\ p)$ was required for fp.

Attacks possible on NTRUEncrypt :

1. Lattice reduction attack
   The most used algorithm for the lattice reduction attack is the Lenstra-Lenstra-Lovàsz algorithm. Because the public key **h** contains both **f** and **g** one can try to obtain them from **h**. It is however too hard to find the secret key when the NTRUEncrypt parameters are chosen secure enough. The lattice reduction attack becomes harder if the dimension of the lattice gets bigger and the shortest vector gets longer.
2. Ciphertext attack
   The chosen ciphertext attack is also a method which recovers the secret key **f** and thereby results in a total break. In this attack adversary tries to obtain his/her own message from the ciphertext and thereby tries to obtain the secret key.

Security and Performance Improvements

Using the latest suggested parameters (see below) the NTRUEncrypt public key cryptosystem is secure to most attacks. There continues however to be a struggle between performance and security. It is hard to improve the security without slowing down the speed, and vice versa.

One way to speed up the process without damaging the effectiveness of the algorithm is to make some changes in the secret key **f**. First, construct **f** such that$f = 1 + pF$, in which **F** is a small polynomial (i.e. coefficients {-1, 0, 1}). By constructing **f** this way, **f** is invertible mod p. In fact $f^{-1} = 1\ mod\ p$, which means that receiver does not have to actually calculate the inverse and that he does not have to conduct the second step of decryption. Therefore constructing **f** this way saves a lot of time but it does not affect the security of the NTRUEncrypt because it is only easier to find f_p but **f** is still hard to recover. In this case **f** has coefficients different from -1, 0 or 1, because of the multiplication by p. Second, **f** can be written as the product of multiple polynomials, such that the polynomials have many zero coefficients. This way fewer calculations have to be conducted.

Chapter 3 Challenges of Post Quantum Cryptography

Some cryptographic systems, such as RSA with a four-thousand-bit key, are believed to resist attacks by large classical computers but do not resist attacks by large quantum computers. Some alternatives, such as McEliece encryption with a four-million-bit key, are believed to resist attacks by large classical computers and attacks by large quantum computers.

If someone announces the successful construction of a large quantum computer fifteen years from now, so are we ready to protect the data. The following are some challenges to post quantum cryptography

This section gives three answers—three important reasons that parts of the cryptographic community are already starting to focus attention on post quantum cryptography:

• We need time to improve the efficiency of post-quantum cryptography.

• We need time to build confidence in post-quantum cryptography.

• We need time to improve the usability of post-quantum cryptography

3.1 Efficiency

Elliptic-curve signature systems with O (b)-bit signatures and O (b)-bit keys appear to provide b bits of security against classical computers. State-of-the art signing algorithms and verification algorithms take time $b^{2+o(1)}$. Can post-quantum public-key signature systems achieve similar levels of performance? My two examples of signature systems certainly don't qualify: one example has signatures of length $b^{2+o(1)}$, and the other example has keys of length $b^{3+o(1)}$. There are many other proposals for post-quantum signature systems, but I have never seen a proposal combining O (b)-bit signatures, O (b)- bit keys, polynomial-time signing, and polynomial-time verification.

Inefficient cryptography is an option for some users but is not an option for a busy Internet server handling tens of thousands of clients each second. If you make a secure web connection today to https://www.google.com, Google redirects your browser to http://www.google.com, deliberately turning off cryptographic protection. Google does have some cryptographically protected web pages but apparently cannot afford to protect its most heavily used web pages. If Google already has trouble with the slowness of today's cryptographic software, surely it will not have less trouble with the slowness of post-quantum cryptographic software.

Constraints on space and time have always posed critical research challenges to cryptographers and will continue to pose critical research challenges to post-quantum cryptographers. On the bright side, research in cryptography has produced many impressive speedups, and one can reasonably hope that increased research efforts in post-quantum cryptography will continue to produce impressive speedups. There has already been progress in several directions.

3.2 Confidence

Merkle's hash-tree public-key signature system and McEliece's hidden-Goppa code public-key encryption system were both proposed thirty years ago and remain essentially unscathed despite extensive cryptanalytic efforts. Many other candidates for hash-based cryptography and code-based cryptography are much newer; multivariate-quadratic cryptography and lattice based cryptography provide an even wider variety of new candidates for post quantum cryptography. Some specific proposals have been broken. Perhaps a new system will be broken as soon as a cryptanalyst takes the time to look at the system.

One could insist on using classic systems that have survived many years of review. But often the user cannot afford the classic systems and is forced to consider newer, smaller, faster systems that take advantage of more recent research into cryptographic efficiency.

To build confidence in these systems the community needs to make sure that cryptanalysts have taken time to search for attacks on the systems. Those cryptanalysts, in turn, need to gain familiarity with post-quantum cryptography and experience with post-quantum cryptanalysis.

3.3 Usability

The RSA public-key cryptosystem started as nothing more than a trapdoor one-way function, "cube modulo n." Modern RSA encryption does not simply cube a message modulo n; it has to first randomize and pad the message. Furthermore, to handle long messages, it encrypts a short random string instead of the message, and uses that random string as a key for a symmetric cipher to encrypt and authenticate the original message. This infrastructure around RSA took many years to develop, with many disasters along the way, padding standard broken by Bleichenbacher in 1998. Furthermore, even if a secure encryption function has

been defined and standardized, it needs software implementations—and perhaps also hardware implementations—suitable for integration into a wide variety of applications.
Implementers need to be careful not only to achieve correctness and speed but also to avoid timing leaks and other side-channel leaks. Post-quantum cryptography, like the rest of cryptography, needs complete hybrid systems and detailed standards and high-speed leak-resistant implementations.

Chapter 4 Conclusion

The need to research and implement this system is very important for future security purpose. Why we care about this now is because of: Long term confidential documents will be readable once quantum computer is build. E.g. military secrets, Electronic signatures on long-term commitments can be forged once quantum computers are available. E.g. No one will inform you that if some secret agency build quantum computer. So it is a precaution to develop a highly secure cryptographic system.

Acknowledgement

I express my profound sense of gratitude to our seminar guide Professor D.C. Jinwala sir, Computer Engineering Dept. SVNIT for allowing me to carry out the seminar work under his eminent guidance and supervision. His consistent support, positive attitude, motivation and the thought provoking suggestions enabled us to complete this seminar successfully.

References

[1] Daniel J. Bernstein (2009). "Introduction to post-quantum cryptography"

[2] Peter W. Shor (1995-08-30). "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer"

[3] Multivariate Cryptography. http://en.wikipedia.org/wiki/Multivariate_cryptography

[4] Lattice Based Cryptography. http://en.wikipedia.org/wiki/Lattice-based_cryptography

[5] Lattice based problems. http://en.wikipedia.org/wiki/Lattice_problems#cite_note-ajtai-18

[6] Merkle Signature scheme. http://en.wikipedia.org/wiki/Merkle_signature_scheme

[7] Ralph Merkle. "Secrecy, authentication and public key systems / A certified digital signature". Ph.D. dissertation, Dept. of Electrical Engineering, Stanford University, 1979.

[8] NTRUEncrypt Algorithm. http://en.wikipedia.org/wiki/NTRUEncrypt