Quantum-Resistant Protocol for Two-Party Communication

CS6903 Spring ‘21 Project 2

Ly Cao, Randy Genere, Faisal Hameed, Kevin Ye

Introduction

- Problem: most public-key algorithms can be broken efficiently by quantum computers
- Their security relies on the hardness of these Math problems:
    - Integer factorization
    - Discrete logarithm
    - Elliptic curve discrete logarithm
- Need: an asymmetric key cryptographic scheme secure against attacks from quantum computers
- Project goal: Implement two secure post-quantum cryptographic algorithms
    - Code-based
    - Lattice-based

Algorithms in NSA Suite B

The NSA considers the following algorithms to be secure enough to protect SECRET and even TOP SECRET information depending on the configuration that is used.

- AES
- SHA2, SHA3
- RSA
- DH, ECDH
- DSA, ECDSA

“Very Powerful” Quantum Computer

- Shor’s Algorithm and Grover’s Algorithm would allow us to break many types of encryption that are currently secure
- Currently a computer doesn’t exist that is sufficiently powerful to run either of these algorithms
    - The most powerful quantum computer created by Google has 53 qubits
    - Factoring a 1024-bit integer using Shor’s algorithm would require more than 2000 qubits
- Breaking 2048-Bit RSA would require a quantum computer with over 4000 qubits which would be considered “very powerful”

Shor’s Algorithm

- RSA relies on the difficulty of the “factoring problem” (factoring the product of two large prime numbers)
- DH and DSA rely on the difficulty of the Discrete Logarithm Problem (determining r such that $r = \log_g x \bmod p$), while ECDH and ECDSA rely on a modified DLP used in Elliptic Curve Cryptography
- Shor’s Algorithm is able to take advantage of parallel computation to solve the factoring problem
- Starting with a random superposition state of two integers and performing a series of Fourier transformations, a new superposition can be found that holds two integers that satisfy an equation
  - Using this method allows Shor’s Algorithm to solve the Discrete Logarithm Problem

Grover’s Algorithm

- Symmetric Encryption requires brute force, so cracking it requires testing the entire keyspace
- Grover’s Algorithm speeds up this process (a n-bit cipher requires $2^{n/2}$ searches)
- This means AES-128 would actually provide 64-bit protection in a post-quantum world
- Since 80 bit is considered secure, AES would only be secure by using 192 bit or 256 bit keys
- By creating a table of $N^{1/3}$ and using Grover’s algorithm, hash functions can also be broken
  - Providing b-bit security would now require a 3b-bit output
- SHA2 and SHA3 with longer outputs still remain quantum resistant

Benefits of Lattice-based and Code-based Cryptography

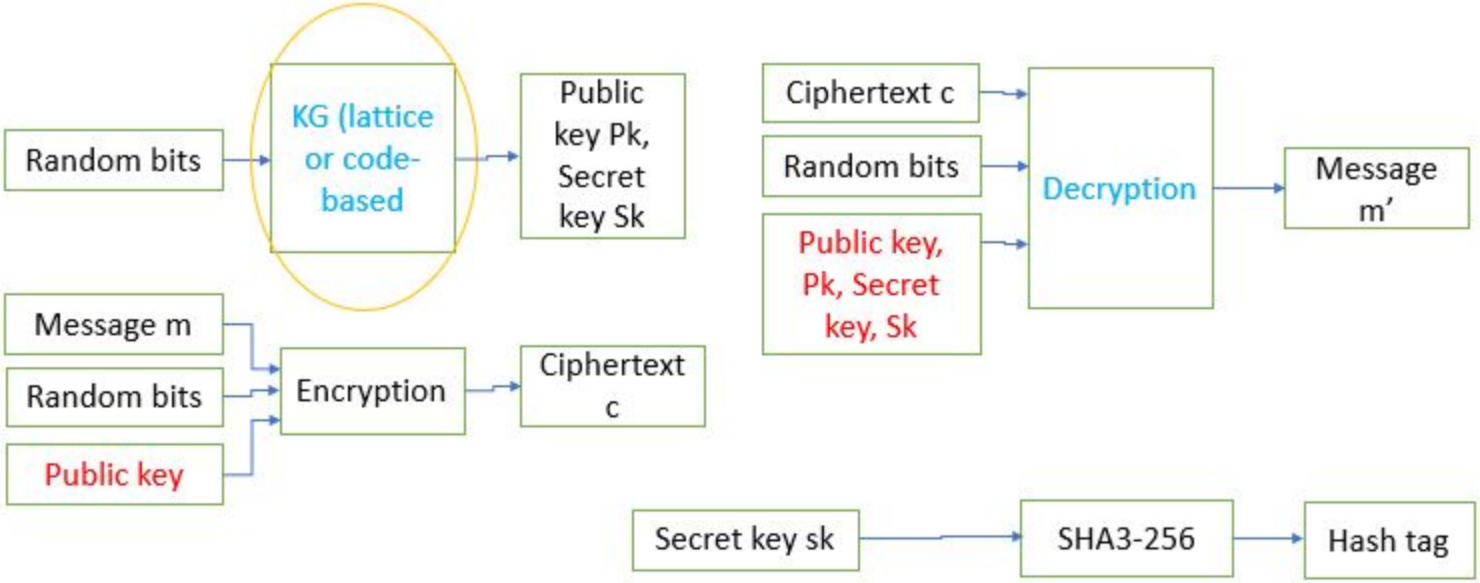
- Avoid the weaknesses of RSA by multiplying matrices instead of multiplying large prime numbers
- Based on the hardness of lattice problems
- NTRU is the most secure and efficient lattice-based scheme
    - Relies on the difficulty of factoring certain polynomials which makes it resistant against Shor's Algorithm

- Algorithms that make use of error correcting codes
- Based on the difficulty of decoding linear codes
- Considered resistant to quantum attacks when key sizes are increase by a factor of 4
- Can be solved by transforming into a Low-Weight-Code-World Problem
    - Solving a LWCWP is not possible in large enough dimensions

Neither makes use of the Hidden Subgroup Problem (factoring and DLP), so they are quantum resistant.

General Design

- IND-CCA KEM either from lattice-based problem or code-based problem to generate symmetric key
- IND-CCA Symmetric Encryption to encrypt and decrypt messages



Lattice-based KEM

I. Background
II. NTRU
III. Connection to Closest Vector Problem
IV. Choice of Parameters

Lattice-based KEM

I. Background:

A lattice is defined by vector space in R^m spanned by a set of n basis vectors with integer coordinates.

$$\mathcal{L}(B) = \{Bx : x_i \in \mathcal{Z}, \forall i = 1, 2, \ldots, n\} \quad B \in R^{m \times n}$$

An example would be

$$b_1 = [{0 \atop 1}], b_2 = [{1 \atop 0}]$$

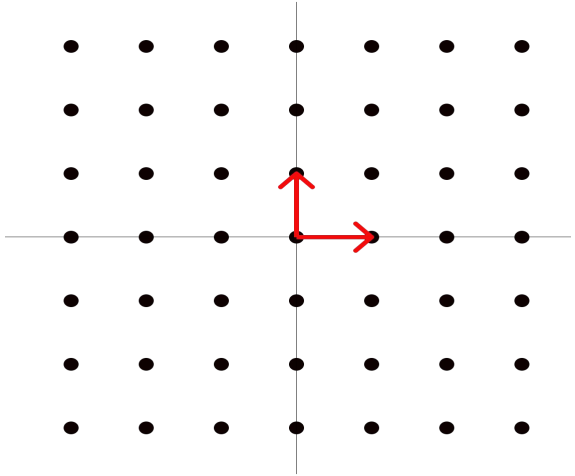
$$v = b_1 + b_2 = [{1 \atop 1}] \in \mathcal{L}(B)) \text{ and } \in span(B)$$

$$v' = 0.1b_1 + b_2 = [{0.1 \atop 1}] \notin \mathcal{L}(B)) \text{ and } \in span(B)$$

Lattice-based KEM

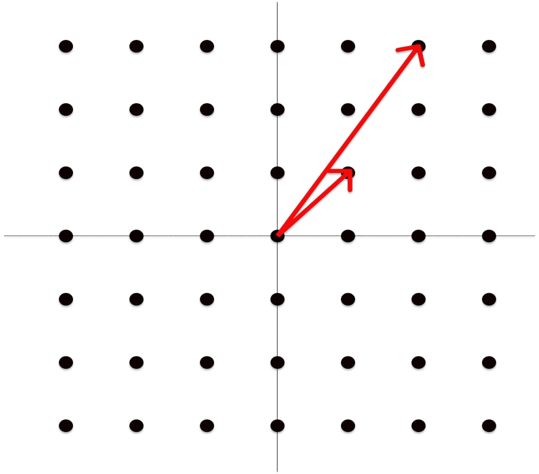
I. Background:

A lattice could be represented by different set of m linearly independent basis vectors in R^n. Below are two geometric examples where two different bases describe the same lattice.



Good basis vectors are as orthogonal as possible

$$b_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, b_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



Bad Bases are not orthogonal

$$b_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b_2 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

Lattice-based KEM

I. Background:

Bad bases make it hard to find a closest vector from the lattice given any vector with real coordinates in span(B).

$$t = \begin{bmatrix} 1.8 \\ 0.2 \end{bmatrix}$$

$$t \in span(B) \notin \mathcal{L}(B), v \in \mathcal{L}(B)$$

Closest vector $\mathrm{v} \in \mathcal{L}(B)$ to t from good basis

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} v = t = \begin{bmatrix} 1.8 \\ 0.2 \end{bmatrix}$$

$$v = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$v = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ in standard coordinate system

Closest vector $\mathrm{v} \in \mathcal{L}(B)$ to t from bad basis

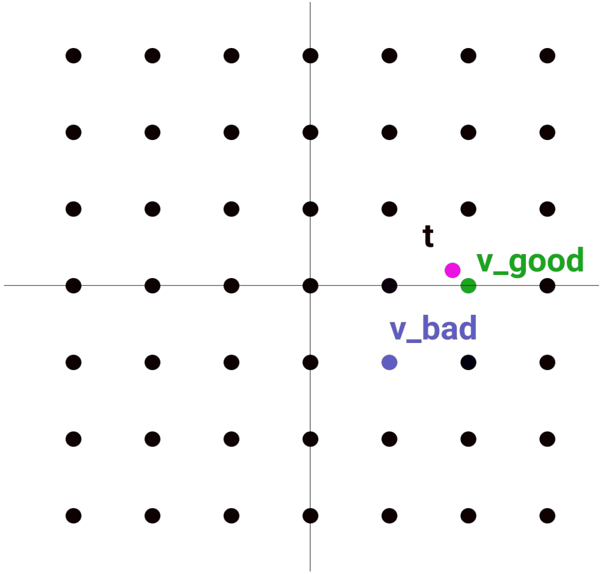
$$B = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix} v = t = \begin{bmatrix} 1.8 \\ 0.2 \end{bmatrix}$$

$$v = \begin{bmatrix} 5 \\ round(-1.6) \end{bmatrix} = \begin{bmatrix} 5 \\ -2 \end{bmatrix}$$

$v = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ in standard coordinate system

Lattice-based KEM

I. Background:



Good basis found

$v = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ in standard coordinate system

Bad basis found

$v = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ in standard coordinate system

- Given a set of bad basis vectors, it's hard to find the set of good basis vectors since traditional method like Gram-Schmidt will result in a new set orthogonal bases that do not define the same lattice as the original bases.
- Closest Integer Vector Problem is hard if we are given a bad basis

Lattice-based KEM

I. Background:

- There are two hard lattice problems we will use: the closest vector problem and shortest vector problem
- Closest Vector Problem
    - Given a vector t in R^m and a lattice L(B), find the closest vector to t in L(B)
- Shortest Vector Problem
    - Given a lattice L(B), find the shortest vector in the lattice

Lattice-based KEM

II. NTRU:

- NTRU is an algorithm that uses the hardness of closest vector problem in a lattice to create an asymmetric cryptosystem where private keys are good bases and public keys are bad bases of the same lattice.
- Bases are polynomial terms, while coefficients define integer coordinates given the base.

Lattice-based KEM

II. NTRU:

- Polynomial ring with coefficients in R

$$R[x] = \{a_0 + a_1x + a_2x^2 + \cdots + a_nx^n : n \geq 0 \text{ and } a_0, a_1, \ldots, a_n \in R\}$$

- Quotient ring of R by m is a set of congruent classes modular m

$$R/(m) = R/mR = \{\bar{a} : a \in R\}, \bar{a} = \{a' : a' = a \pmod{m}\}$$

- Ring of convolution polynomials
- Ring of Convolution

Lattice-based KEM

II. NTRU:

- Polynomial ring with coefficients in R

$$R[x] = \{a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n : n \geq 0 \text{ and } a_0, a_1, \ldots, a_n \in R\}$$

- Quotient ring of R by m is a set of congruent classes modular m

$$R/(m) = R/mR = \{\bar{a} : a \in R\}, \bar{a} = \{a' : a' = a \pmod{m}\}$$

- Ring of convolution polynomials $R = \frac{Z[x]}{x^N - 1}$
- Ring of convolution polynomials (modular q) $R_q = \frac{Z/qZ[x]}{x^N - 1}$

Lattice-based KEM

II. NTRU:

- Product of two polynomials in R

$$a(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{N-1} x^{N-1} \in R$$

vector of coefficients is defined as $(a_0, a_1, a_2, \ldots, a_{N-1}) \in Z^N$

b(x) and c(x) are defined similarly

$$a(x) \times b(x) = c(x), c_k = \sum_{i+j=k \text{ (mod N)}} a_i b_{k-i}$$

- Define Lift (or center lyft) operation for a polynomial a(x) in Rq is bringing the coefficients a_i to the range [-p/2, p/2] while preserving its modular q
- Ternary (trinary polynomial):
  - T(d1,d2) = a(x) in R where a(x) has d1 coefficients equal 1, d2 coefficients equal -1, and other coefficients equal 0

Lattice-based KEM

II. NTRU:

- Parameters: (N, p,q, d), q > (6d+1)p
- Private keys:
  - f(x) in T(d+1,d) , g(x) inT(d,d) where coefficients of f(x), g(x) are either -1,0,or 1
  - Fq(x) = f(x) $^{-1}$ in Rq
  - Fp(x) = f(x) $^{-1}$ in Rp
- Public key: h(x) = Fq(x) * g(x) (mod q) in Rq
- Message m(x) in R where coefficients are in (-p/2, p/2] (center lyft m(x) in R to be a polynomial in Rp)
- Ciphertext c(x) are encoded in Rq
- Gen (both Alice and Bob):
  - Randomly generate private keys f(x), g(x)
  - Some random private bytes s for hash function in decryption failure case
  - Compute Fq(x), Fp(x)

Lattice-based KEM

II. NTRU:

- Encryption:
  - Alice uses Bob's public key h(x) to compute c(x)
  - Choose a random function r(x) in T(d,d)
  - Randomly generate a message m(x) in Rp
  - Return c(x) = p h(x) * r(x) + m(x) (mod q)
  - Send c(x) to Bob
  - Sym_key (symmetric key) = Hash(m(x)) (HASH = SHA3-256)
- Decryption:
  - Bob uses his private keys f(x), Fp(x)
  - a(x) = f(x) * c(x) (mod q), center lyft a(x)
  - b(x) = Fp(x) * a(x) (mod p) (b(x) = m(x))
  - Return Sym_key = Hash(b(x))
  - On failure return Hash(c(x), s)

Lattice-based KEM

II. NTRU:

- The scheme is IND-CCA secure. Proof is found by Saito, Xagawa, and Yamakawa in reference 2
- Secure against quantum attack due to the hardness of closest vector problem and shortest vector problem in a lattice

Lattice-based KEM

III. Connection to Lattice Problem

1) For key recovery, attacker has to solve the shortest vector problem in a NTRU lattice to recover f(x), g(x)
    - From Reference 1, page 427, the vector (f,g) is shorter than Gaussian heuristic prediction of the shortest vector in a lattice by a factor O(1/sqrt(N)), so there is a high chance that (f,g) are very short vectors in NTRU lattice
    - h(x) is a large vector since even though f(x) is short, Fq(x) tends to be large (page 420, reference 1)

Lattice-based KEM

III. Connection to Lattice Problem

2) For plaintext recovery, attacker has to solve the closest vector problem in a NTRU lattice to recover m(x)

- Given public key h, define the lattice (reference 5)
- $L_h = \{(a, b) \in Z^{2N} : a * h \equiv b (\bmod q)\}$
- (pr(x)) * h(x) (mod q) is a vector in the lattice
- m(x) (mod q) is a very short vector
- Recovering m(x) is equivalent to finding the closest vector in lattice L_h and subtract it from c(x)

- There are no known efficient solutions to the 2 problems

Lattice-based KEM

IV. Parameter Choice:

We follow the parameter choices and implementation of submission ntruhps4096821 to NIST post-quantum standardization due to its superior performance

N = 821, q= 2^{12}, p=3, d = 820

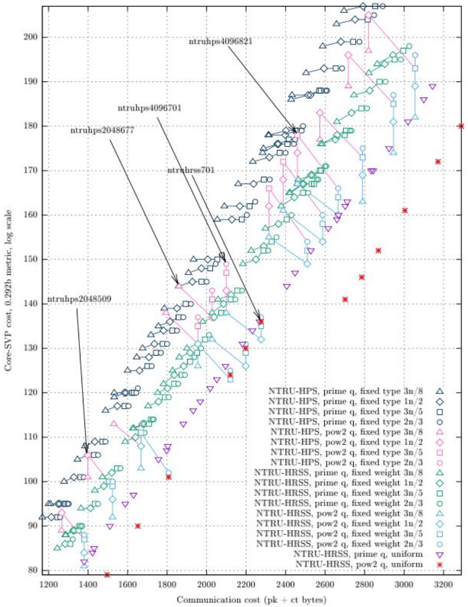


Fig. 9: Size vs. security trade-offs as described in Section 6. Lines connect parameter sets that use the same n. All of the parameter sets use $p = 3$. The "fixed type d" parameter sets take $\mathcal{L}_g = \mathcal{L}_m = \mathcal{T}_0(d)$ and $\mathcal{L}_f = \mathcal{L}_r = \mathcal{T}$. The "fixed weight d" parameter sets take $\mathcal{L}_f = \mathcal{L}_g = \mathcal{T}_+(d)$, $\mathcal{L}_r = \mathcal{T}(d)$, and $\mathcal{L}_m = \mathcal{T}$. All parameter sets are clean, correct, and use the smallest q available. Security is evaluated using a Core-SVP model.

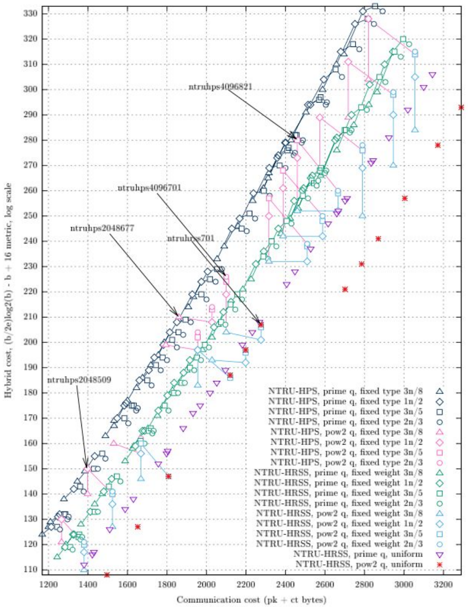
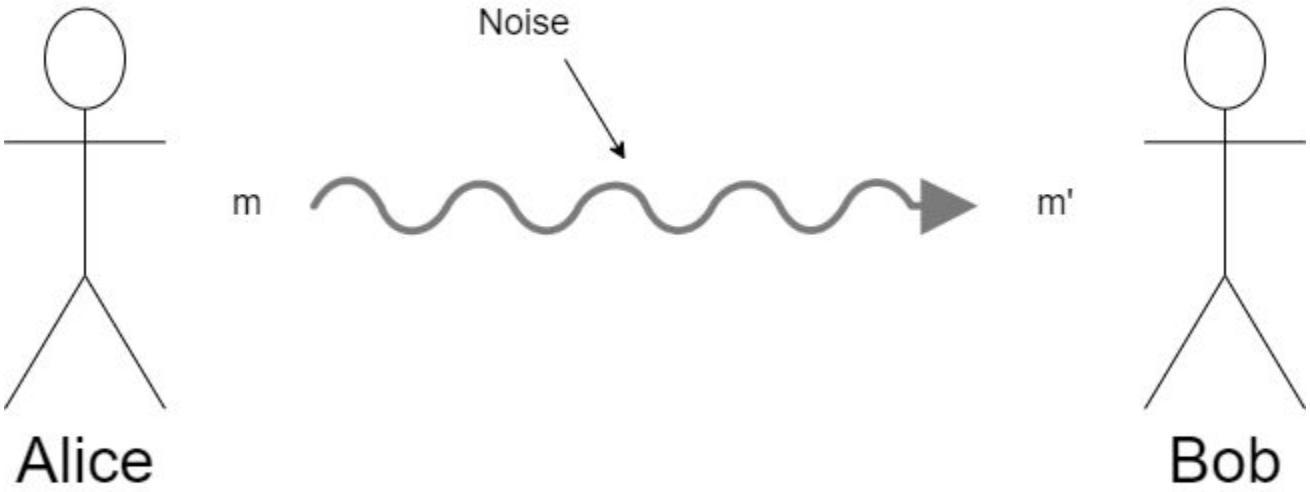


Fig. 10: Size vs. security trade-offs as described in Section 6. Lines connect parameter sets that use the same n. All of the parameter sets use $p = 3$. The "fixed type d" parameter sets take $\mathcal{L}_g = \mathcal{L}_m = \mathcal{T}_0(d)$ and $\mathcal{L}_f = \mathcal{L}_r = \mathcal{T}$. The "fixed weight d" parameter sets take $\mathcal{L}_f = \mathcal{L}_g = \mathcal{T}_+(d)$, $\mathcal{L}_r = \mathcal{T}(d)$, and $\mathcal{L}_m = \mathcal{T}$. All parameter sets are clean, correct, and use the smallest q available. Security is evaluated with respect to the hybrid attack.

Ntruhps4096821 performance from reference 3

Code-based KEM

Background: Error-correcting linear codes

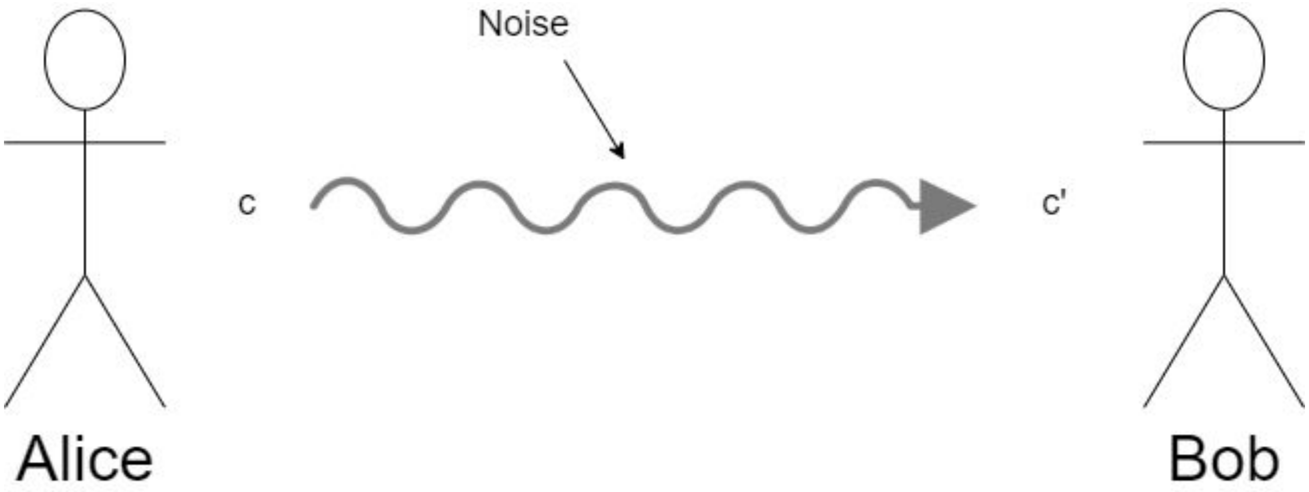


Alice sends m through a noisy channel that garbles the message and produces m’

How can Bob recover m from m’?

Code-based KEM

Background: Error-correcting linear codes



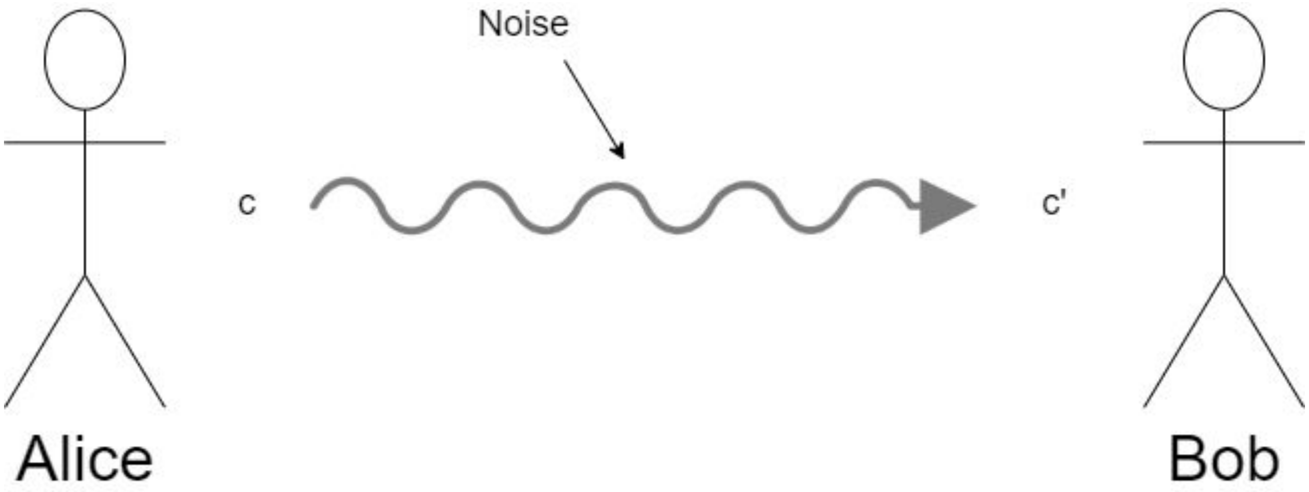
Solution: Alice encodes m into a codeword c correctable up to t errors

There is a unique c reachable from a c’ that contains up to t errors

Code-based KEM

Simple example: Hamming code

Background: Error-correcting linear codes



Solution: Alice encodes m into a codeword c correctable up to t errors

There is a unique c reachable from a c’ that contains up to t errors

Code-based KEM

Background: Error-correcting linear codes

The **Hamming distance**, (or distance), between two strings x and y is the number of positions i such that $x(i) \neq y(i)$

Let C be the set of codewords in a code

The **minimum distance** of a code is the minimum distance between any two c1 and c2, such that $c1, c2 \in C$

The code with minimum distance d is correctable up to $\lfloor d-1/2 \rfloor$ errors

Code-based KEM

Background: Error-correcting linear codes

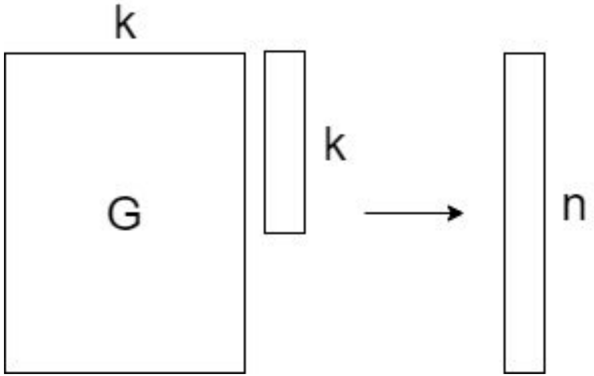
A code represented by C which is a subspace of a finite field $\mathbb{F}_q^n$ s a **linear code**

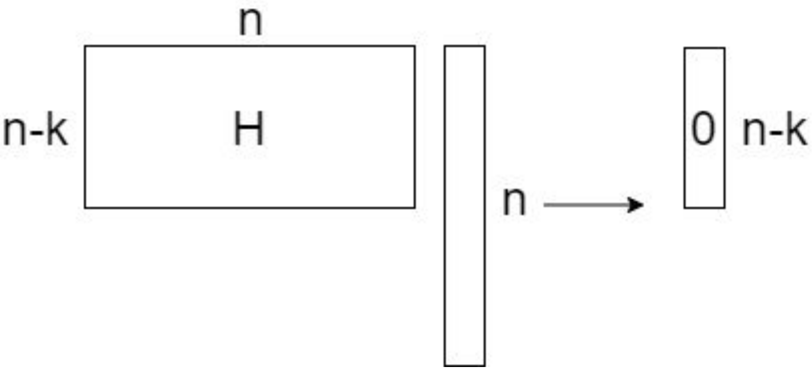
Encoding map: $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$

Map messages of k-length into codewords of n-length, n>k

Code-based KEM

Background: Error-correcting linear codes





G is a generator matrix for code C if C is the column span of G

H is a parity-check matrix for C if for all c such that Hc = 0, c is in C

Code-based KEM

Idea: Code-based cryptography

It is **computationally hard** to decode a codeword for some arbitrary code

but…

Some well-structured codes have efficient decoding algorithms -> trapdoor

Code-based KEM

Idea: McEliece cryptosystem

Bob’s secret:

- generator matrix G, efficiently decodable up to t errors
- invertible scrambler matrix S
- permutation matrix P

Bob’s public key: PGS

PGS looks like random generator matrix. Sender computes PGSm and generates error vector e with up to weight t (weight = number of nonzeros). Send PGSm + e

Bob inverts P to get G(Sm) + P^{-1}e. Bob can decode G to get Sm, and then invert S to get m

Code-based KEM

Idea: Neiderreiter cryptosystem

Bob's secret:

- Parity check matrix H, efficiently decodable up to t errors
- invertible scrambler matrix S
- permutation matrix P

Bob's public key: SHP

Sender sends SHPe where e is an error vector with up to weight t

Bob inverts S to get G(Pe). Bob can decode G to get Pe, and then invert P to get e

Code-based KEM

NIST submission

A KEM based on a variant of the Neiderreiter system using binary Goppa codes was submitted to Round 3 of NIST's Post-Quantum Cryptography Standardization competition.

- Irreducible polynomial g(x) with degree t over F_q, $q = 2^m$
- A sequence of q distinct elements $(a_1,...,a_q)$ over F_q
- The binary Goppa code is characterized by $(g, a_1, a_2,..., a_n)$

We are using a parameter set from the submission such that:

m = 13, n = 6688, t = 128

According to the authors, a McEliece-based public key would need between roughly 300KB and 1.5MB length to achieve an effective key length between 128 and 256 bits.

Our chosen parameter set produces a 1MB public key

Symmetric Encryption

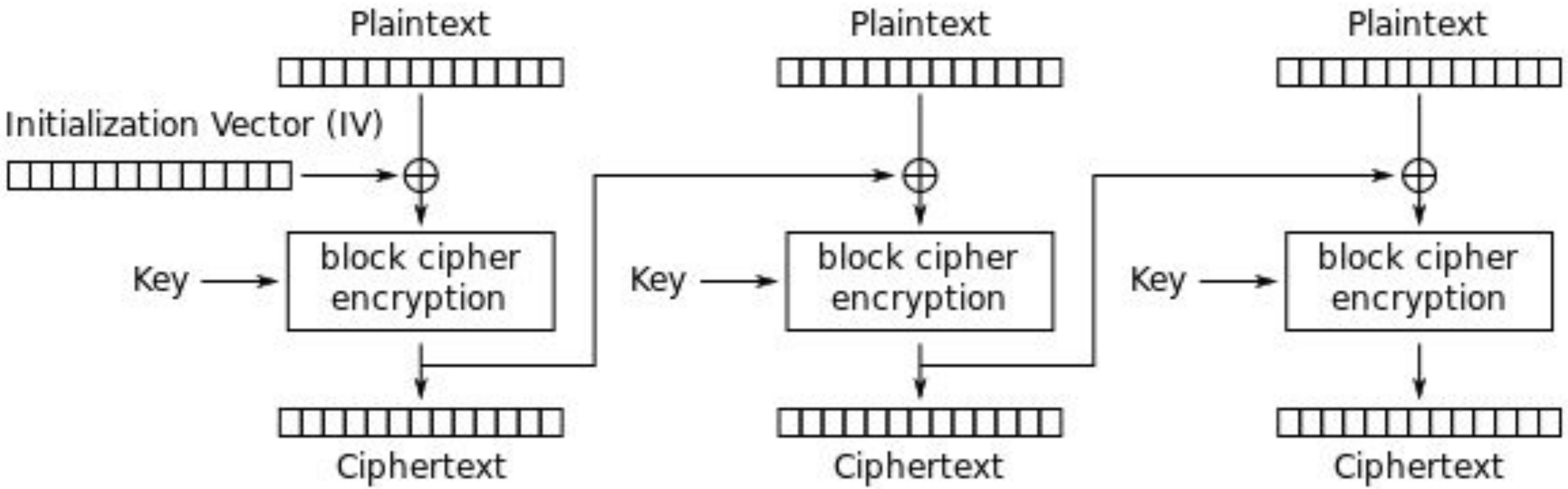
Gen:

- Symmetric key from KEM
- For each session, generate a random start sequence number seq that increments cyclically (go back to 0 when reaches maximum) for every message sent out. Call this operation Cyc_increment

Encryption for each message:

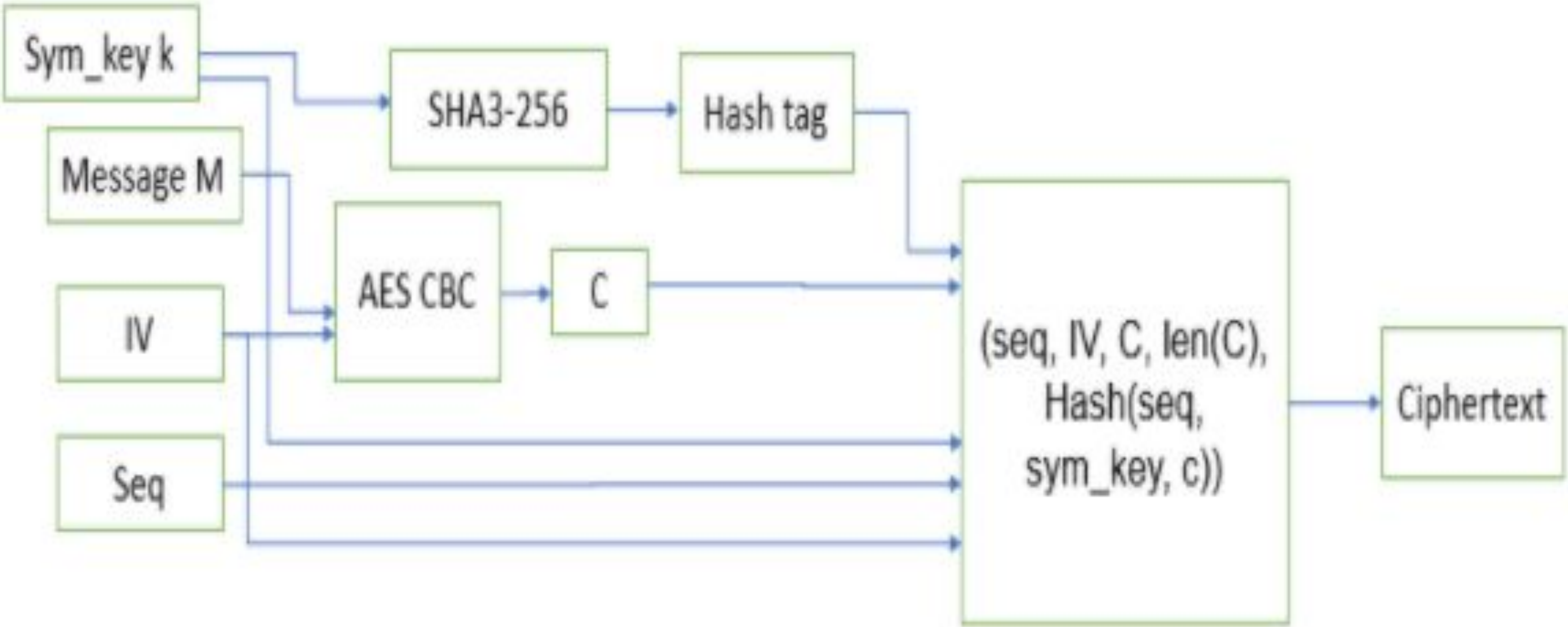
- Enc = AES-256 in CBC mode
- Hash = SHA3-256
- Generate random IV
- C = Enc(IV, message)
- Ciphertext = (seq, IV, C, len(C), Hash(seq, sym_key, c))
- Seq = Cyc_increment(seq)

AES - 256 in CBC Mode [r.6]



Cipher Block Chaining (CBC) mode encryption

Encryption

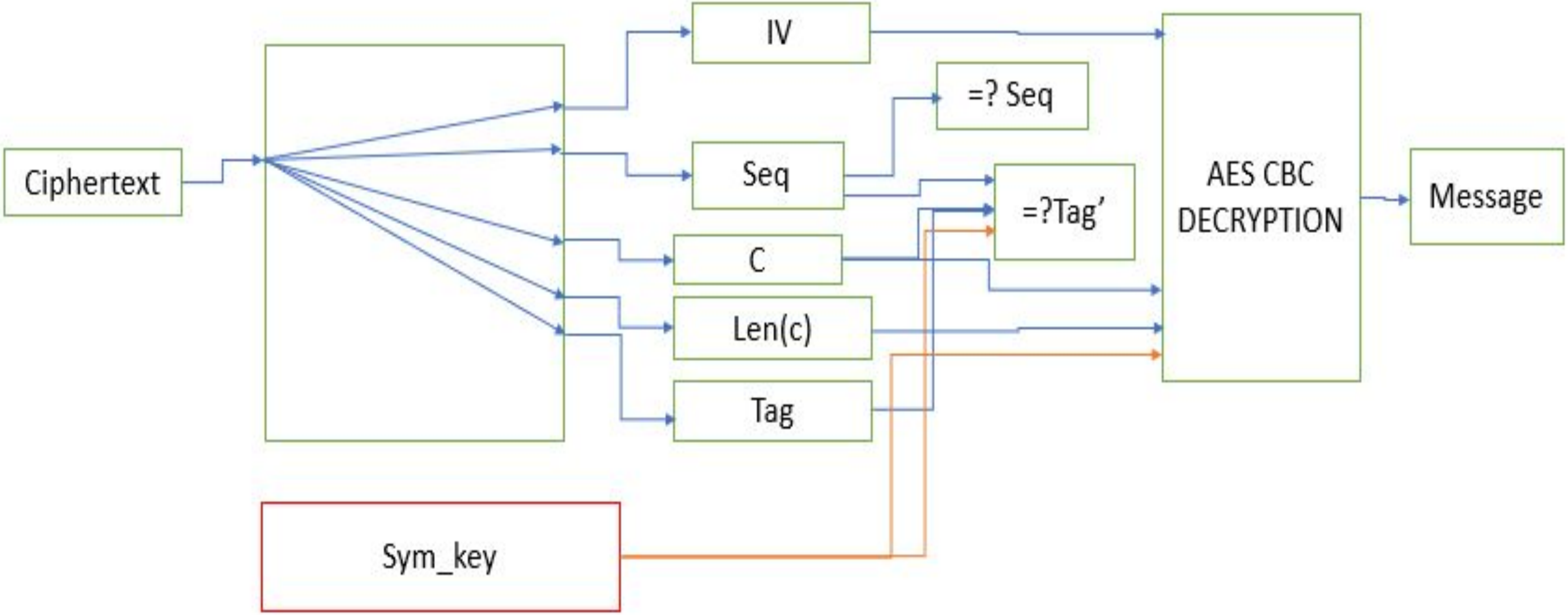

Sym_key k
Message M
IV
Seq
SHA3-256
Hash tag
AES CBC
C
(seq, IV, C, len(C), Hash(seq, sym_key, c))
Ciphertext

Symmetric Encryption

Decryption:

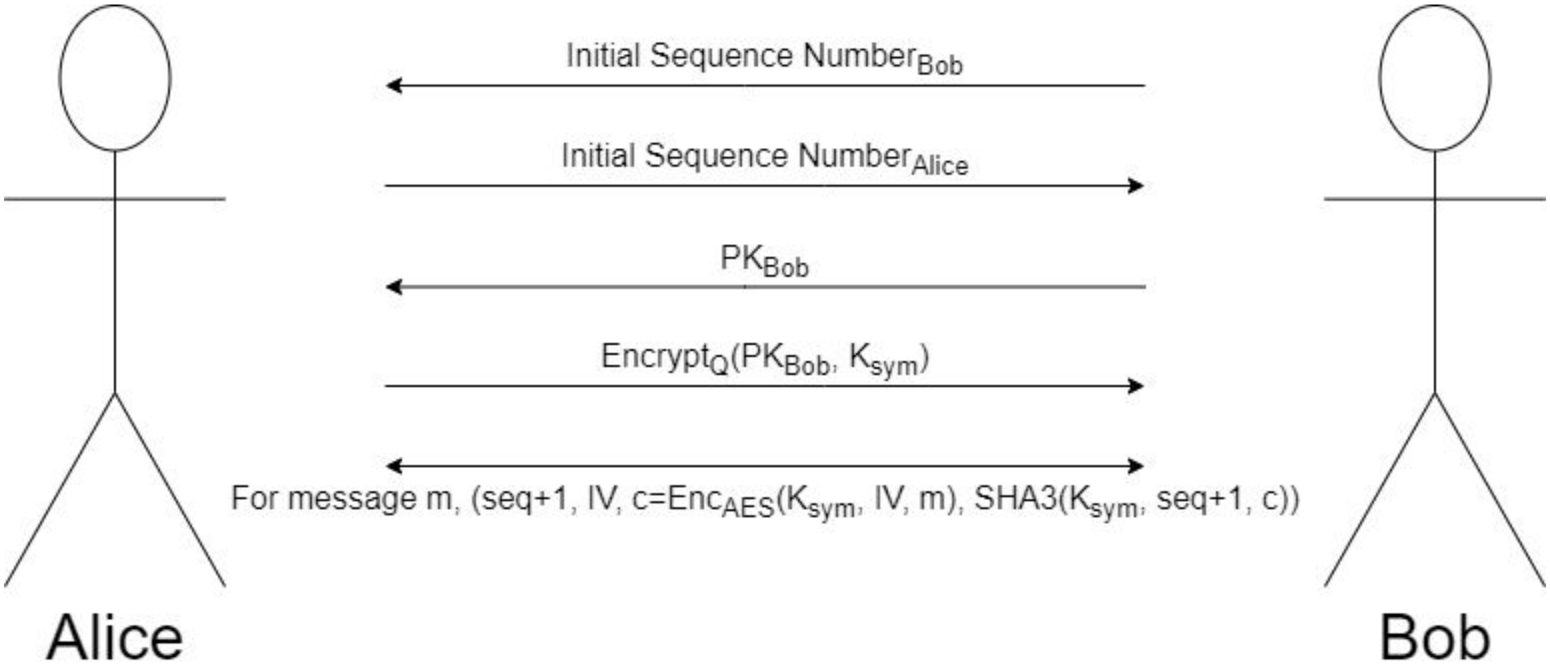
- On ciphertext, extract seq, IV, c, len(c), Tag
- Check if seq is the expected sequence from the other party
- Compute Tag' = Hash(seq, sym_key, c) and Verify that Tag' = Tag
- Dec = Decryption algorithm of AES-256 in CBC mode
- Message = Dec(IV, sym_key, c, len(c))

Decryption



IV
=? Seq
Ciphertext
Seq
=?Tag'
AES CBC
DECRYPTION
Message
C
Len(c)
Tag
Sym_key

Two-Party Quantum-Resistant Hybrid Scheme



Performance and Measurements

	NTRU	McEliece
Key length (pk, sk)	1230 bytes, 1590 bytes	~1MB, ~13.6KB
Ciphertext length	1230 bytes (328 byte message)	240 bytes (836 byte message)
Sender runtime	~1.5ms	~1ms
Receiver runtime	~13ms	~160ms

References

1) An Introduction to Mathematical Cryptography (2014) - Hoffstein, Pipher, Silverman, chapter 2 and chapter 7
2) https://eprint.iacr.org/2017/1005.pdf
3) https://eprint.iacr.org/2018/1174.pdf
4) https://ntru.org/resources.shtml
5) http://math.stmarys-ca.edu/wp-content/uploads/2017/07/Ahsan-Zahid.pdf
6) https://classic.mceliece.org/nist/mceliece-20201010.pdf
7) https://www.youtube.com/watch?v=qisORKNShvo (on Goppa codes)
8) https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation