



INDIAN STATISTICAL INSTITUTE KOLKATA

A SURVEY OF NIST PQC CODE-BASED PROPOSALS : CLASSIC McELIECE AND BIKE

SK RIJWAN

Work done under the supervision of

Dr. Shion Samadder Chaudhury

Department of Cryptography, IAI, TCG CREST and

Dr. Anirban Ghatak

Applied Statistics Unit, ISI Kolkata

JULY 4, 2024

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
M.TECH. CRYPTOLOGY AND SECURITY

Declaration

I, SK Rijwan (Roll No: CrS2214), hereby declare that this report entitled **A SURVEY OF NIST PQC CODE-BASED PROPOSALS : CLASSIC MCELIECE AND BIKE**, submitted to the Indian Statistical Institute, Kolkata, towards the fulfillment of the requirements for the degree of Master of Technology in Cryptology and Security, is an original work carried out by me under the supervision of **Dr. Shion Samadder Chaudhury** [Department of Cryptography, IAI, TCG CREST] and **Dr. Anirban Ghatak** [Applied Statistics Unit, ISI Kolkata] . This report has not formed the basis for the award of any degree or diploma in this or any other institution or university. Whenever a piece of external information, statement, or result has been used, it has been duly acknowledged and cited.

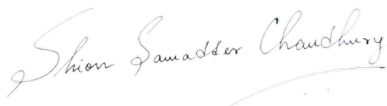
SK Rijwan

Roll No: CrS2214

Date:03/07/2024

Acknowledgements

I thank everyone who has assisted me in seeing this project through to its completion. I would like to first express my profound gratitude and deepest regards to my Guides and ISI Kolkata, and sincerely wish to acknowledge their vision, guidance, valuable feedback and constant support throughout this project. I am indebted to my friends for their steadfast encouragement and time. I am lastly grateful to the Indian Statistical Institute, Kolkata for providing the necessary resources and facilities to complete this project to the best of my ability.



Signature of primary supervisor

Shion Samadder Chaudhury,
Assistant Professor,
Department of Cryptology,
Institute for Advancing Intelligence,
TCG Centres for Research and Education in Science and Technology.



Signature of secondary supervisor

Dr. Anirban Ghatak ,
Applied Statistics Unit,
ISI Kolkata .

Contents

1	Introduction - Chapter	1
2	Preliminaries	6
2.1	Linear Codes	6
2.2	Goppa Codes	7
2.3	General Decoding Problem	9
2.3.1	Syndrome Decoding Problem	9
2.4	Patterson's Algorithm	9
2.4.1	Correction of errors/Syndrome decoding of Goppa codes	9
2.4.2	Patterson's Algorithm for Error Correction	11
2.5	QC-MDPC Codes	12
2.6	Decoding of QC-MDPC Codes:	13
2.6.1	Bit Flipping(BF) Algorithm:	14
2.7	Estimation of Decoding failure rate:	15
2.8	Hard Quasi-Cyclic Computational Problems	16
2.8.1	Decisional QCSD:- Quasi-Cyclic Syndrome decoding problem	16
2.8.2	Decisional QCCF:- Quasi-Cyclic codeword finding problem	17
2.8.3	Utilizing the Quasi-Cyclic Structure	17
3	The Classic McEliece : conservative Code-based cryptography	18
3.1	The McEliece Code-based Cryptosystem	18
3.1.1	Introduction	18

3.1.2	Outline of the McEliece CBC	19
3.1.3	Attacks on the McEliece CBC	20
3.2	A Brief Review of Information Set Decoding Attacks	20
3.3	One-wayness of the McEliece CBC	25
3.3.1	CCA Vulnerability of McEliece - Sloppy Alice Attacks	28
3.4	Niederreiter Cryptosystem	29
3.4.1	Introduction	29
3.4.2	Outline of the Niederreiter cryptosystem	29
3.5	The Classic McEliece PKE/KEM	31
3.5.1	The Classic McEliece Parameters:	32
3.5.2	Design:	32
3.5.3	Cryptanalysis	34
3.6	Partial Information Attack on Classic McEliece	35
4	BIKE	39
4.1	Introduction	39
4.2	Instance of Niederreiter scheme with QC-MDPC	40
4.3	Overview of BIKE	40
4.3.1	Specification	40
4.3.2	BIKE KEM	42
4.3.3	Decoder	43
4.3.4	Pseudo-random Bits Generation	46
4.4	Security of BIKE	46
4.4.1	OW-CPA Security of BIKE:	47
5	Conclusion	49

1 | Introduction - Chapter

In recent years, a significant change is seen in the topics of interest in the field of Cryptography. Previously, the focus was primarily on the long-established public-key cryptosystems like the well-known Rivest-Shamir-Adleman (RSA) (64) algorithm and algorithms based on elliptic curve cryptography (ECC) (65).

RSA Algorithm

- Given $n = pq$, the product of two large primes, and a positive integer b with $\gcd(b, \phi(n)) = 1$.
- Select $e > 1$, encryption exponent s.t. $\gcd(e, \phi(n)) = 1$ where $\phi(n) = (p - 1)(q - 1)$.
- Compute $d = e^{-1} \pmod{\phi(n)}$; the decryption exponent.
- Public key (n, e) ; private key (n, d) .
- Encryption: Compute ciphertext $\beta = \alpha^e$ for message $\alpha \in Z_n$.
- Decryption: A computes $\gamma = \beta^d$ to get back $\alpha^{ed} = \alpha$.

The two major strategies attacking RSA cryptosystem are

- Factorize n to compute $\phi(n)$
- Find d (decryption exponent) directly.

Among these two strategies the factoring attack has been most intensively pursued. The problem for this attack is known as *integer factorization problem*: Given an integer n which is a product of two distinct prime numbers p, q ($n = p \cdot q$), factorize n to find p and q . When p and q are sufficiently large and chosen properly, the factorization is

pretty hard and there is still no polynomial time algorithm to solve the *integer factorization problem*.

Some of the *classical algorithms* (The term 'classical algorithm' has been used to differentiate with the quantum algorithms, which is discussed later in this chapter) for the *integer factorization problem* are

- Fermat's difference of squares method;
Euler's extension of Fermat's method.
- Pollard's $p - 1$ algorithm
- Pollard's Rho algorithm
- The Quadratic Sieve
- The General Number Field Sieve (NFS)

Among these algorithms NFS (66) has proved to be the most efficient with subexponential complexity in the size of the number to be factored and weakened the security of the RSA cryptosystem.

In the mean time another public-key cryptosystem based on the elliptic curve cryptography(ECC) emerged with smaller key sizes compared to RSA while preserving the same level of security. The security of ECC based cyptosystems is based on the hardness of the *discrete logarithm problem*: find an integer a (if it exists) such that $g^a = h$ when given $g, h \in G$, where G is a finite group. There is still no polynomial time algorithm known to solve the *discrete logarithm problem*.

So, after decades of research the RSA and ECC based cryptosystems provides sufficient computational security against the classical attacking algorithms.

There are another type of algorithms ,called *quantum algorithms* which are designed to run on a quantum computer. Quantum computer uses *qubits* which is analogous to classical bits . Qubits may assume both values- 0, 1 simultaneously during computation. It follows that with n qubits , 2^n states can exist simultaneously indicating the possibility of using *Quantum logic gates* during computation, resulting

in reduction of time complexity with respect to classical computers.

The major classes of quantum computing are

1. The Hidden Subgroup Problem(HSP)
2. Search algorithms (Grover's algorithm)
3. Quantum system simulations.

Among these algorithms HSP and Grover's (67) algorithm pose a real threat to the security of the existing public-key cryptosystems.

In (1994) Peter Shor (68) proposed an algorithm aiming to factor composite numbers into their prime number components using Euler's method. Shor's algorithm can be interpreted as an Order-finding algorithm, which is an instance of the HSP. Shor's algorithm solves the *integer factorization problem* and the *discrete logarithm problem* on polynomial time when run on a quantum computer. Though there is uncertainty about the advent and development of quantum computers in the near future, it is better to have precaution against the possibility of development of a powerful quantum computer breaking the security of currently used public-key cryptosystems. Thus the field of post quantum cryptography(PQC) arose and captured the interest of many researchers.

In 2017, the National Institute of Standards and Technology (NIST) initiated the standardization of quantum-resistant public-key cryptographic algorithms through the NIST Post-Quantum Cryptography (PQC) project. Submissions were divided into two categories: key encapsulation mechanisms (KEM) and digital signatures. Initially 69 candidates were selected for the first round(2017). After the completion of the first round in 2019 , 26 of those candidates were selected to proceed for the second round. 15 of those were selected for the third round(2020). 7 of those were considered as finalist and 8 were selected as alternate candidates. After the end of third round (2022) 4 candidates(one KEM and three digital signatures) were selected for standardization and 4 (all KEMs) were selected to proceed for the fourth round .The KEM selected for standardization is CRYSTALS-KYBER (69) and the digital

signatures to be standardized are CRYSTALS–Dilithium (70), FALCON (76), and SPHINCS⁺ (77). The alternate key-establishment candidate algorithms : BIKE (72), Classic McEliece (7), HQC (71), and SIKE (78) were considered for future standardization and advanced to round four for evaluation. In the same year SIKE was proved to be not secure by a successful attack on it. The 5th NIST PQC Standardization Conference took place in 2024 aimed to discuss on various elements of the algorithms under selection and evaluation, as well as to gather insightful feedback to guide standardization decisions. NIST extended invitations to the teams behind BIKE, Classic McEliece, Falcon, and HQC to provide updates on their respective algorithms.

CRYSTALS–KYBER is a lattice-based KEM while BIKE, Classic McEliece, HQC are code-based KEMs. While lattice-based KEMs have efficient computation and moderate key sizes ,existence of some inherent structure give rises to uncertainties about their security in the future. The code based cryptosystems have relatively larger key sizes but have a long history of security, well-studied theoretical foundation.

Classic McEliece is a code-based KEM that uses a binary Goppa code in the Niederreiter (8) variant of the McEliece cryptosystem (1) combined with standard techniques to achieve CCA security. Based on the assumption that the 1978 McEliece scheme provides one-way under chosen-plaintext attacks (OW-CPA) security a tight proof of the submitted KEM’s IND-CCA2 security in the quantum random oracle model is given in various research results and their submission paper to the NIST. The fourth round submission is a merger of the second-round submissions Classic McEliece and NTS-KEM. NIST expresses confidence in the security of Classic McEliece and may consider to standardize it.

BIKE(Bit Flipping Key Encapsulation) BIKE differs from Classic McEliece in its choice of family codes used. While Classic McEliece relies on binary Goppa codes, BIKE utilizes QC-MDPC (Quasi-Cyclic Moderate-Density Parity-Check) codes, inspired by the Niederreiter cryptosystem. Both BIKE and Classic McEliece are based

on the Niederreiter cryptosystem, where the main challenge is decoding random linear codes. This similarity allows BIKE to also demonstrate CPA-security. As BIKE employs a different family of codes, it utilizes a different decoding algorithm (Black Gray Flip decoder) compared to Classic McEliece. The decryption method is associated with chances of decryption failure unlike Classic McEliece. Though the decryption failure rate (DFR) is low. Initially entered with 3 variants, BIKE has only one variant in the final submission with claims of being chosen ciphertext attack (CCA) secure. BIKE also remains to be a suitable choice for standardization by NIST.

HQC (Hamming Quasi-Cyclic) also uses QC-MDPC like BIKE but does not use Niederreiter template. Unlike Classic McEliece and BIKE, HQC does not hide any code structure and is more complex in its design (Analogous with some lattice scheme working on code-based environment).

Though HQC also is a tough competitor in NIST PQC, due to the differences with Classic McEliece and BIKE, only the survey of the later two KEMs are included in the report.

2 | Preliminaries

2.1 Linear Codes

The concept of error correcting codes come from the problem of sending information over a noisy channel. A t Error correcting code can successfully decode the original message sent if there is a occurrence of maximum t errors .

Linear codes are error correcting codes with the property that the sum of any two codewords is also a codeword.

Let F_q be a finite field with q elements. A $[n, k]$ code C is a linear code of length n and dimension k , i.e., a k dimensional subspace of F_q^n . Every element of the code C is called a codeword c .

The Hamming weight denoted by $w_H(c)$ of an element $c \in F_q^n$ is the number of nonzero entries of c . Hamming distance of two codewords $y, c \in F_q^n$ is $d(y, c) = w_H(y - c)$.

A generator matrix G for the code C is a $k \times n$ matrix such that $C = mG : m \in F_q^k$. So G corresponds to a map sending a message of length k to a element of F_q^n .

A parity check matrix H for the code C is a $(n - k) \times n$ matrix such that $C = c \in F_q^n : Hc^T = 0$.

Let $x = c + e$ is transmitted where $w_H(e) \leq t$. Then c is the unique codeword closest to x and the term $s(x) = Hx^T = H(c^T + e^T) = He^T$ is called the *syndrome* of x .

A systematic generator matrix is of the form $(I_k | P)$ where I_k is the the $k \times k$ identity matrix and P is $k \times (n - k)$ matrix. Then the parity check matrix is of the form $(P^T | I_{n-k})$.

2.2 Goppa Codes

Let $L = (\alpha_1, \dots, \alpha_n) \in \mathbb{F}_{2^m}$ be a set of points known as Goppa points and $g(x) \in \mathbb{F}_{2^m}[x]$ be an irreducible, degree- t polynomial known as Goppa polynomial. Then we define a Goppa code (16)

$$C(L, g) = \left\{ c \in \mathbb{F}_2^n : \sum_{i=1}^n \frac{c_i}{x - \alpha_i} \equiv 0 \pmod{g(x)} \right\}. \quad (1)$$

This code can correct upto t errors.

Let us consider $\hat{H}_{\text{Goppa}}(L, g) \in \mathbb{F}_{2^m}^{t \times n}$ of the form

$$\hat{H}_{\text{Goppa}}(L, g) = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1} & \alpha_2^{t-1} & \cdots & \alpha_n^{t-1} \end{pmatrix} \cdot \begin{pmatrix} g^{-1}(\alpha_1) & 0 & \cdots & 0 \\ 0 & g^{-1}(\alpha_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g^{-1}(\alpha_n) \end{pmatrix}.$$

From $\hat{H}_{\text{Goppa}}(L, g) \in \mathbb{F}_{2^m}^{t \times n}$, we construct the parity-check matrix

$H_{\text{Goppa}}(L, g) \in \mathbb{F}_2^{mt \times n}$ by applying the bijection $V : \mathbb{F}_{2^m} \rightarrow \mathbb{F}_2^m$, that represents \mathbb{F}_{2^m} as an m -dimensional vector space over \mathbb{F}_2 , i.e., $\sum_{i=0}^{m-1} a_i \gamma^i \mapsto [a_0, \dots, a_{m-1}]$.

Complexity Classes in Computer Science

In computer science, problems are classified according to their computational complexity. This classification helps in understanding how difficult it is to solve a problem and how time is required as the input size grows. Here are the some classes of problems:

- **P (Polynomial Time):**
 - Problems that can be solved by a deterministic Turing machine in polynomial time. In other words, there exists an algorithm that can solve

the problem in time $O(n^k)$ for some constant k , where n is the size of the input.

- **Examples:** Sorting a list (e.g., Merge Sort), finding the greatest common divisor (Euclidean algorithm).

- **NP (Nondeterministic Polynomial Time):**

- Problems for which a solution can be verified in polynomial time by a deterministic Turing machine. If a given solution can be checked quickly (in polynomial time), the problem belongs to NP.
- **Examples:** Boolean satisfiability problem (SAT), Hamiltonian path problem.

- **NP-complete:**

- These are the hardest problems in NP. A problem is NP-complete if it is in NP and as hard as any problem in NP, meaning any NP problem can be transformed into it using a polynomial-time reduction.
- **Examples:** SAT (Satisfiability), 3-SAT, Traveling Salesman Problem (decision version).

- **NP-hard:**

- Problems that are at least as hard as the hardest problems in NP. However, NP-hard problems are not necessarily in NP (they may not have solutions verifiable in polynomial time).
- **Examples:** Halting problem, Generalized Chess (determining if a player can guarantee a win from a given position).

2.3 General Decoding Problem

2.3.1 Syndrome Decoding Problem

on **Input:** $H \in \mathbb{F}_2^{(n-k) \times n}$, $s \in \mathbb{F}_2^{n-k}$, integer t Can one find $e \in \mathbb{F}_2^n$ such that $w_H(e) \leq t$ and $eH^T = s$?

The general decoding problem for a linear code involves either solving for the syndrome or the codeword c itself, without any assumption other than that c satisfies: $cH^T = 0$. This general decoding problem and the problem of finding the weight distribution of a code (i.e., finding the numbers of codewords of given weights) were proved to be NP-complete by Berlekamp, McEliece, and van Tilborg in 1978(2).

2.4 Patterson's Algorithm

Let $\Gamma(L, g(x))$ be a Goppa code, where $g(x)$ is a primitive polynomial with $\deg(g(x)) = t$ and $|L| = n$. Let $\dim_{\mathbb{F}_q}(\Gamma(L, g(x))) = k$, and G be $k \times n$ sized generator matrix for respective Goppa code; then encoding of a k -length message vector m over \mathbb{F}_q is mG .

2.4.1 Correction of errors/Syndrome decoding of Goppa codes

Let the vector $y = (y_1, y_2, \dots, y_n)$ be received with r number of errors, where $2r + 1 \leq d'$ (for maximum number of error correction). Let $L = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$,

$$y = (y_1, y_2, \dots, y_n) = (c_1, c_2, \dots, c_n) + (e_1, e_2, \dots, e_n)$$

with $e_i \neq 0$ at exactly r -places. We need to

- locate positions of error (say $B = \{i : 1 \leq i \leq n \text{ and } e_i \neq 0\}$);
- find the corresponding error values (values of $e_i : i \in B$).

In order to find these, we define two polynomials

[Error locator polynomial $\sigma(z)$ and Error evaluator polynomial $w(z)$]

$$\sigma(z) = \prod_{i \in B} (z - \alpha_i) \quad (\text{this is a 'r' degree polynomial});$$

$$w(z) = \sum_{i \in B} e_i \beta_i \prod_{j \in B, j \neq i} (z - \alpha_j) \quad (\text{this is a 'r - 1' degree polynomial}).$$

[Syndrome of received vector] The Syndrome of received vector y is defined as $S(y)$

where:

$$S(y) = \sum_{i=1}^n \frac{y_i}{x - \alpha_i} = \sum_{i \in B} \frac{e_i}{x - \alpha_i} \pmod{g(x)}.$$

Error-correction: Algorithm for correcting $r \leq \left\lfloor \frac{d'}{2} \right\rfloor$ errors in a Goppa code:

Step (i): Compute the syndrome

$$S(y) = \sum_{i=1}^n \frac{y_i}{x - \alpha_i}$$

Step (ii): Solve the key equation

$$\sigma(x)S(y) = w(x) \pmod{g(x)}$$

by writing

$$\sigma(x) = x^r + \sigma_{r-1}x^{r-1} + \cdots + \sigma_1x + \sigma_0,$$

$$w(x) = w_{r-1}x^{r-1} + \cdots + w_1x + w_0$$

and solve for r equations and $2r$ unknowns. If the code is binary, take

$$w(x) = w_0.$$

Step (iii): Determine the set of error locations

$$B = \{i : 1 \leq i \leq n \text{ and } \sigma(\alpha_i) = 0\}.$$

Step (iv): Compute the error values $e_i = \frac{w(\alpha_i)}{\sigma'(\alpha_i)}$ for all $i \in B$.

Step (v): The error vector $e = (e_1, e_2, \dots, e_n)$ is defined by e_i for $i \in B$ and zeros elsewhere.

Step (vi): The codeword sent is calculated as $c = y - e$.

2.4.2 Patterson's Algorithm for Error Correction

The Patterson algorithm (17) decodes only binary Goppa codes. It computes the syndrome $S(y)$ of a received vector y . It then solves the key equation $\sigma(x)S(y) = w(x) \pmod{g(x)}$ by heavily exploiting the requirement that the code is binary. The error locator polynomial can be split in even and odd powers of x such that $\sigma^2(x) + x^2\tau^2(x^2) = 0$, as field has characteristic 2.

The Patterson algorithm can be described as below:

Input: The received vector y and the Goppa code $\Gamma(L, g)$.

Step (i): Compute syndrome $S(y)$ an element of F_q .

Step (ii): Compute $T(x) = S(y)^2 \pmod{g(x)}$.

Step (iii): Compute $P(x) = \sqrt{T(x)} \pmod{g(x)}$.

Step (iv): Compute and solve $\tau(x)$ with $x\tau(x^2) = P(x) \pmod{g(x)}$.

Step (v): Compute the locator polynomial $\sigma(x) = x\tau(x^2)$.

Step (vi): Find the roots of $\sigma(x)$.

Step (vii): Compute the error vector e .

Output: The error vector e .

With the error vector the codeword can be recovered from the erroneous vector received (that is the codeword plus random error with weight up to t).

2.5 QC-MDPC Codes

Quasi-Cyclic(QC) Code : A linear code \mathcal{C} of length $n = n_0 + n_1$ is called a quasi-cyclic code if the cyclic shift of any codeword by n_0 symbols yields another codeword.

The generator and the parity-check matrix of the quasi-cyclic code \mathcal{C} is completely defined by its first row because every other row is a cyclic shift by n_0 symbols of the preceding one.

circulant matrix: A circulant matrix is a square matrix in which every row is a cyclic shift of the previous one and every column is a cyclic shift of the adjacent column.

A circulant matrix A :

$$A = \begin{pmatrix} a_0 & a_{n-1} & a_{n-2} & \cdots & a_1 \\ a_1 & a_0 & a_{n-1} & \cdots & a_2 \\ a_2 & a_1 & a_0 & \cdots & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{pmatrix}$$

is fully determined by its first row. The polynomial $a(x) = \sum_{i=0}^{n-1} a_i x^i$ associated with the first row of $A = (a_0, a_1, \dots, a_{n-1})$ is called the *generating polynomial* of A . A cyclic shift of a row corresponds to the multiplication of the polynomial $a(x)$ by $x \pmod{(x^n - 1)}$. Therefore, there is a natural one-to-one correspondence between circulant matrices of size $r \times r$ and ideals of the quotient ring $F_2[x]/(x^r - 1)$.

The product of two circulant matrices is a circulant matrix. Furthermore, the product of two circulant matrices corresponds to the multiplication of their generating polynomials modulo $(x^r - 1)$. The inverse of a non-singular circulant matrix A is also a circulant matrix, and the polynomial corresponding to A^{-1} is the inverse of the

polynomial $a(x)$ in the ring $\mathbb{F}_q[x]/(x^r - 1)$.

The parameter r defining the ring \mathbb{R} used in BIKE is chosen such that the polynomial defining \mathbb{R} factors as $(x^r - 1) = \phi_r(x)(x - 1) \in F_2[x]$, where the cyclotomic polynomial $\phi_r(x) = (x^r - 1)/(x - 1) \in F_2[x]$ is irreducible. As a result the checking of the invertibility of the elements of the polynomial ring \mathbb{R} is easy. And it follows that only the elements with odd weight are invertible. $\phi_r(x)$ is irreducible when r is prime and 2 is a primitive root modulo r .

(QC-MDPC code). An (n_0, k_0, r, w) -QC-MDPC code is an (n_0, k_0) quasi-cyclic code of length $n = n_0r$, dimension $k = k_0r$, order r (and thus index n_0), admitting a parity-check matrix with constant row weight $w = O(\sqrt{n})$.

The sparsity of the parity-check matrix allows the use of relatively efficient iterative decoding techniques, such as the bit-flipping algorithm. Bit-flipping decoding is a method of choice in BIKE because it offers good properties while also being very simple. BIKE makes use of $(2, 1)$ -QC codes. Such codes are subspaces of \mathbb{R}^2 . The private key $(h_0, h_1) \in \mathcal{H}_w$ defines the code

$$C = \{(fh_1, fh_0) \mid f \in \mathbb{R}\} = \{(f_0, f_1) \in \mathbb{R}^2 \mid f_0h_0 + f_1h_1 = 0\}$$

with generator and parity check matrices (in $\mathbb{R}^{1 \times 2}$) respectively:

$$G = \begin{bmatrix} h_1 & h_0 \end{bmatrix}, \quad H = \begin{bmatrix} h_0 \\ h_1 \end{bmatrix}.$$

The corresponding binary matrices are obtained by expanding the polynomials into circulant block.

2.6 Decoding of QC-MDPC Codes:

Decoding of QC-MDPC codes can be achieved by iterative decoders with bit flipping (BF) algorithm (which is also used for decoding LDPC codes). But BF-based

decoders are probabilistic decoders with non-zero decoding failure rate (DFR). DFR, which is defined as the percentage of decoding failures in a given number of decoding attempts, affects the efficiency and security of cryptographic schemes based on QC-MDPC codes. Several improvements have been made for BF-based decoders with negligible DFR.

2.6.1 Bit Flipping (BF) Algorithm:

The Bit Flipping algorithm was proposed by Gallager (75) in 1963. The Bit-Flipping (BF) algorithm is described using the Tanner graph of an LDPC/MDPC code. The Tanner graph provides a visual representation of the parity-check matrix and comprises two sets of nodes:

1. **Check nodes** (illustrated by squares in Fig. 3) represent the rows of the parity-check matrix. Therefore, every Tanner graph has $N - K$ check nodes (each representing a row of the matrix).
2. **Variable (Bit) nodes** (illustrated by circles in Fig. 3) represent the N bits of a received vector that should be decoded. Each bit node in the Tanner graph corresponds to a column in the parity-check matrix, as the size of the matrix is given by the number of columns and N .

The following steps are performed to generate the Tanner graph of an LDPC/MDPC code with the $N - K$ parity-check matrix H in which h_{ij} is the element located at row i and column j :

1. Draw $N - K$ check nodes (squares) and N bit nodes (circles).
2. Connect each check node to bit node j if $h_{ij} = 1$.

The number of edges that are connected to the bit nodes (i.e., the column weight) indicates the number of codedword bits participating in the i -th parity-check equation (defined by row i of H).

The BF decoding algorithm then operates as follows:

1. We generate the Tanner graph of the code using the parity-check matrix H .
2. Given a specific vector $\mathbf{r} = (r_1, r_2, \dots, r_N)$ that should be decoded, label each bit node j of the Tanner graph with r_j ($1 \leq j \leq N$).
3. We compute the result of the parity-check equation for every row H as

$$S_i = \sum_{j=1}^N h_{ij}r_j \pmod{2} \text{ for } 1 \leq i \leq N - K.$$
4. If $S_i = 0 \forall 1 \leq i \leq N - K$, we return \mathbf{r} as the decoded word.
5. Otherwise, we label the check node i with "unsatisfied".
6. We compute another label f_j for each bit node j , where f_j is the number of "unsatisfied" check nodes connected to bit node j .
7. Flip the value of r_j if f_j is greater than a predetermined threshold, T .
8. Set the error limit Iter for iterations.
9. If $1 \leq i \leq N - K$ such that $S_i \neq 0$, we return "failure" and terminate the algorithm.
10. With the updated \mathbf{r} , we jump to step 2.

This algorithm is probabilistic with non-zero DFR.

2.7 Estimation of Decoding failure rate:

Decoding of BIKE is not a deterministic algorithm unlike BIKE. There is a chance of not being able to decrypt at all, which is referred to as decryption failure. The decryption failure rate (DFR) is a key parameter for the performance and CCA security of BIKE. The technique for estimation of DFR for BIKE is listed below as given in the specification provided in the BIKE round 4 submission.

The Low Impact of Block Size on Computational Assumptions

The block size r must be chosen large enough to allow efficient decoding. In practice, one must choose $r = \Omega(wt)$. The higher r , the lower the Decoding Failure Rate (DFR). On the other hand, the best known attacks for codes of rate $\frac{1}{2}$ are of order $2^{t(1+o(1))}$ or $2^{w(1+o(1))}$. This is corrected by a polynomial factor in r , which is very small in practice. An interesting consequence is that if w and t are fixed, a moderate modification of r (say plus or minus 50%) will not significantly affect the resistance against the best known key and message attacks. This simplifies the extrapolation methodology described in the next paragraph.

Estimating the DFR by Extrapolation

Low DFRs, e.g., 2^{-128} , as required for Chosen-Ciphertext Attack (CCA) security, cannot be directly estimated by simulation. Instead, simulations are combined with extrapolations, as described next. First, the DFR is measured for smaller block sizes r , for which simulations provide reliable estimations. Subsequently, one can define a curve based on the sample of r -DFR acquired values, and the curve is extrapolated to a larger block size where the extrapolated DFR reaches the target. Linear extrapolation over two acquired points tends to overestimate the required r (i.e., a conservative estimation). More extensive simulations can refine the DFR estimation and hence lead to smaller (more desirable) sufficient r .

2.8 Hard Quasi-Cyclic Computational Problems

2.8.1 Decisional QCSD:- Quasi-Cyclic Syndrome decoding problem

Given $h \in \mathbb{R}$, a vector $y \in \mathbb{R}$, and target $t > 0$, determine whether there exists $(e_0, e_1) \in \mathbb{R}^2$ such that $|e_0| + |e_1| = t$ and $e_0 + e_1 h = y$.

2.8.2 Decisional QCCF:- Quasi-Cyclic codeword finding problem

Given $h \in \mathbb{R}$ and target $v > 0$, determine whether there exists $(c_0, c_1) \in \mathbb{R}^2$ such that $|c_0| + |c_1| = v$ and $c_0 + c_1 h = 0$.

Though there is a search to decision reduction in the context of the general syndrome decoding problem, for the quasi-cyclic case, no such reduction is known. Best known solvers for the quasi-cyclic problems mentioned above derive from Information Set Decoding (ISD), designed for search problems.

2.8.3 Utilizing the Quasi-Cyclic Structure

Identifying codewords and decoding are somewhat simpler (by a polynomial factor) when the target code is quasi-cyclic. If a QC code contains a word of weight w , then its quasi-cyclic shifts are also included. This property typically offers a factor of r speedup compared to a random code. Furthermore, using the Decoding One Out of Many (DOOM) method (79), it is feasible to produce r equivalent decoding instances. Addressing these instances together yields a workload reduction factor of \sqrt{r} .

3 | The Classic McEliece : conservative Code-based cryptography

Classic McEliece is a submission to NIST’s Post-Quantum Cryptography (PQC) Standardization Project (7). Based on the Niederreiter variant (8) of the Goppa-code(16) based McEliece *code-based cryptography* (CBC)(1), the Classic McEliece *Key-encapsulation Mechanism*(KEM)(7) was one of the fourth-round finalists in the public-key encryption/key establishment category of the NIST Post-Quantum standardization [<https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions>].

we will first have some brief description of the McEliece CBC and Niederreiter’s dual version of McEliece CBC. Then have a walk through the design and cryptanalysis of the Classic McEliece *public key encryption*(PKE)/KEM.

3.1 The McEliece Code-based Cryptosystem

3.1.1 Introduction

R.J.McEliece proposed a public key cryptosystem based on algebraic coding theory in 1978 using some Goppa codes (1).Thus the first *code-based cryptosystem*(CBC) was created. This cryptosystem used the hardness of the *random decoding problem*, which was proved to be NP-complete(as mentioned



Figure 3.1:
R.J.McEliece
(21.05.1942 -
08.05.2019)

above in the context of general decoding problem). But there exist fast decoding algorithm (Algorithm due to Patterson) (17) for Goppa codes.

3.1.2 Outline of the McEliece CBC

The Public key:

- Let G' be the generator matrix of an $[n, k]$ irreducible Goppa code Γ , which can correct t errors.
- Let S be a random $k \times k$ invertible matrix and P , a random $n \times n$ permutation matrix.
- Define $G := SG'P \in \mathbb{F}_2^{k \times n}$.
- The tuple (G, t) is the public key.

The Secret key:

- The triple (G', S, P) is the secret key.

Encryption:

- Let $m \in \mathbb{F}_2^k$ be the plaintext, and $e \in \mathbb{F}_2^n$ be a random vector with $w_H(e) \leq t$.
- The ciphertext is defined as: $y = mG + e$. Clearly y is seen as a noisy “codeword” corresponding to the randomized “generator matrix” G .

Decryption:

- Compute $y' = yP^{-1} = mSG' + eP^{-1}$.
- As $w_H(eP^{-1}) \leq t$, the decoder for the t -error correcting Goppa code Γ can decode y' to obtain mS .
- Extract plaintext m using S^{-1} .

The first McEliece proposal used an irreducible binary Goppa code with $n = 1024$, $k = 524$, correcting $t = 50$ errors. The public key was a randomized version of the

code - difficult to decode. The size of the public key was ≈ 32 KB, with a ciphertext of size 1024 bits. The Encryption and Decryption algorithms can be implemented using digital logic and communication rates near 10^6 bits per second is feasible.

3.1.3 Attacks on the McEliece CBC

From the above mentioned design, an attacker who got hold of an encrypted message y has two possibilities in order to retrieve the original message m :

Structural attack: The attacker can try to recover G from G' so that he/she can use Patterson's algorithm. This attack is hopeless if n and t are large enough, because there are huge possibilities for G only and there are possibilities for S and P also. Among other codes with efficient decoding, Goppa codes have resisted structural attacks so far.

Generic attack: The attacker can try to recover m from y without knowing G . Which is same as solving general decoding problem for a random $[n, k]$ code in the presence of at most t errors.

Among the generic attacks structure-oblivious Information Set Decoding (ISD) attacks, which have been considered to be the benchmark for cryptanalysis of McEliece-type CBC's. One of the most successful generic attacks was the modified ISD attack by Bernstein, Lange and Peters in 2008 (5).

3.2 A Brief Review of Information Set Decoding Attacks

The concept of ISD was introduced by E. Prange(4) in relation to decoding cyclic codes. Recalling the decoding problem of random linear codes, upon receiving a input $y \in F_2^n$ we have to find the unique closest codeword $c \in C$, where $y = c + e$, where $w_H(e) \leq t$. If the syndrome decoding problem of x can be solved, i.e., find e s.t. $w_H(e) = w \leq t$ and $He^T = Hx^T$, then the decoding problem can be solved. It follows that e defines the unique linear combination of exactly w columns of H , having the sum He^T . If that w out of n columns can be found, the *syndrome decoding problem* can

be solved. The idea is to find a set I of w vectors whose combination reaches the target value $s(x)$. The search space for linear codes is large enough to be exploited by brute force search. Different ISD algorithms has tried to reduce this search space over the years. Let us now have a brief description of *information sets* before looking into various ISD algorithms.

It is evident that using a systematic generator matrix $G_s = [I_k | P_{k \times (n-k)}]$ implies that the first k symbols of a codeword $c = mG_s$ are information symbols.

Let $G_{k \times n}$ be random generator matrix. We can find a set I with k elements such that $I \subseteq \{1, 2, \dots, n\}$ and the sub-matrix of G formed by these k elements indexed columns, denoted G_I , is invertible. Also, It can be shown that $G_I^{-1}G$ and G are equivalent generator matrices. The I -indexed entries of any codeword $c = mG_I^{-1}G$ are called *information symbols* of c and so, I is an *information set*. The matrix $G_I^{-1}G$ is often termed a ‘systematized’ version of G .

ISD in its simplest form

- takes input $y \in \mathbb{F}_2^n$ with $d(y, C) := \min\{d(y, c) \mid c \in C\} = w = w_H(e)$; let $c \in C$ be the codeword closest to y .
- Assume y and c coincide on the indices of some information set I . Then all these are assumed to be error-free coordinates.
- It follows that $y_I G_I^{-1}$ is the preimage of c under the map induced by G , i.e., $c = (y_I G_I^{-1})G$.

For codes over \mathbb{F}_2 , an equivalent formulation is to identify w columns of the ‘systematized’ H matrix which are chosen by the non-zero entries of the syndrome s . In Prange’s algorithm, this identification is done by randomly choosing w columns, out of (a subset of) n columns of H , in every iteration and checking if their XOR matches the syndrome. One of the main objectives of subsequent developments was to improve the search for the columns of the parity check matrix that correspond to the error vector.

Lee–Brickell’s algorithm: A modification of the basic formulation further reducing

the search space is the Lee Brickell's algorithm (18) which is considered as a classical information set decoding algorithm. The algorithm is as follows

Let p be an integer with $0 \leq p \leq w$

1. Choose an information set I .
2. Replace y by $y - y_I G_I^{-1} G$.
3. For each size- p subset $A = \{a_1, \dots, a_p\} \subseteq I$ and for each $m = (m_1, \dots, m_p)$ in $(\mathbb{F}_q^*)^p$, compute

$$e = y - \sum_{i=1}^p m_i g_{a_i}.$$

If e has weight w print e . Else go back to Step 1.

in step 1, I has to be chosen such a way that G_I is invertible. Step 3 requires going through all possible weighted sums of p rows of G which needs to be subtracted from $y - y_I G_I^{-1} G$ in order to make up for the p errors permitted in I . So its better to not set p very large. $p = 2$ is optimal. If in the first iteration, the chosen I does not yield a weight w codeword, another iteration has to be done.

Stern's Algorithm: Originally proposed for the problem of finding low weight codeword, Stern's attack (19) can be used to decode linear codes by finding low weight codeword(e , as mentioned earlier in the context of ISD) in a slightly larger code.

Stern's algorithm uses the idea of Lee and Brickell to allow a fixed number of errors in the information set. On input of weight of the codeword w and a $(n - k) \times n$ parity check matrix H of a $[n, k]$ linear code over F_2 , Stern randomly selects $n - k$ columns of the parity check matrix and selects a random size l subset Z from those $n - k$ columns. Then he divides the remaining k columns randomly into two sets X and Y . Then he searches for a codeword having $p, p, 0$ nonzero bits in X, Y, Z respectively and $w - 2p$ in the remaining columns. Here p and l are parameters of the algorithm. If the search fails then he starts with a new selection of columns. The searching part of Stern's algorithm has three steps as mentioned in the Bernstein, Lange and Peter's

paper (5).

1. Stern applies elementary row operations on H to make an identity matrix by the $n - k$ selected columns. The algorithm restarts if it fails due to the $(n - k) \times (n - k)$ submatrix of original H not being invertible. To avoid the cost of restarting, he chose each column adaptively as a result of pivot in the previous columns leaving a room for bias with no significant effects in the performance.
2. Each column of the $(n - k) \times (n - k)$ submatrix corresponds to a row where the column has 1 in it. So, l columns of Z corresponds to a set of l rows. For every size- p subset A and B of X and Y respectively, he adds each of these l rows with columns of A and B separately, obtaining l bit vectors $\pi(A)$ and $\pi(B)$.
3. for each collision $\pi(A) = \pi(B)$, Stern computes the sum of the $2p$ columns in $A \cup B$. This sum is an $(n - k)$ -bit vector. If the sum has weight $w - 2p$, Stern obtains 0 by adding the corresponding $w - 2p$ columns in the $(n - k) \times (n - k)$ submatrix. Those $w - 2p$ columns, together with A and B , form a codeword of weight w .

Summary of Bernstein-Lange-Peters'(BLP) attack: One of the most successful fastest known attacks is the modified ISD attack by Daniel J. Bernstein, Tanja Lange, and Christiane Peters in 2008(5). The attack strategy is mostly some improvements over Stern's attack. The modifications are as follows.

- *Reuse of existing pivots:* Instead of applying Gauss elimination to each newly supplied H for same code to get the identity submatrix(as in the Stern's attack), this attack uses the identity submatrix generated in the previous iteration. As about $(n - k)^2 / n$ columns will match, the work for these columns is eliminated.
- *Forcing more pivots:* $(n - k - c)$ columns can be artificially reused and the other c columns can be selected from the remaining k columns. c is a parameter of the algorithm. Choosing extreme value of c like $c = 1$ minimizes the time for

Gaussian elimination but maximizes the number of iterations of the entire algorithm. So a more optimized choice of c is values between 1 and $(k/n)(n - k)$ (as used in the Stern's attack).

- *Faster pivoting*: For Stern's algorithm it has frequently been observed[REF from BLP1] that 25 percent of the rows will have both first and second row added to it. Almost half of this work can be saved by pre-computing the sum of the first two rows(which is only one vector addition).
- *Multiple choices of Z*: Differing from the Stern's attack, instead of selecting one Z this attack selects m number of size- l disjoint subsets Z_1, Z_2, \dots, Z_m with the same X, Y . Though there is an m fold increase in the cost in the first and second step of Stern's algorithm due to this generalization, the chance of finding any particular weight- w word grows by a factor of nearly m because Gaussian elimination which depends on only X and Y is done only once.
- *Reusing additions of the $'$ -bit vectors*: In Stern's algorithm it is observed that while computing $\pi(A)$ or $\pi(B)$ almost $p - 2$ of the $p - 1$ additions were carried out before. This attack(BLT) caches those $p - 2$ additions to reduce the cost of computing $\pi(A)$ or $\pi(B)$. This is very significant as p grows.
- *Faster additions after collision*: In the third step of Stern's algorithm, after finding one collision for the pair (A, B) all the columns of $A \cup B$ are added. Bernstein, Lange, Peters pointed out that there is overlap in many of these additions and not all the additions are necessary. As mentioned in their paper after computing $2(w - 2p + 1)$ rows one already has, on average, $w - 2p + 1$ errors and as soon as the number of errors exceeds $w - 2p$, the pair (A, B) can be safely aborted.

They have successfully implemented an attack against the cryptosystem parameters originally proposed by McEliece by decoding 50 errors in a $[1024, 524]$ code over F_2 in just 1400 days by a single 2.4GHz Core 2 Quad CPU, breaking all the previous records of attack on the McEliece cryptosystem.

Some current ISD developments: In 2011 Bernstein-Lange-Peters another ISD technique called *ball-collision decoding* further modifying the running time to $\mathcal{O}(2^{0.0558n})$ (20).

In 2011 Alexander May, Alexander Meurer, Enrico Thomae present a new algorithm (21) for decoding linear codes that is inspired by a representation technique due to Howgrave-Graham and Joux in the context of subset sum algorithm (23) bringing down the complexity to $\mathcal{O}(2^{0.05363n})$.

In 2012 Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer proposed a new information set decoding algorithm[(22)] with further increment in the number of representations and reducing the time complexity to $\mathcal{O}(2^{0.0494n})$.

3.3 One-wayness of the McEliece CBC

One-wayness means that it is easy to generate ciphertext from input but difficult to invert the map from random input to ciphertext, given the public key. One-wayness is modernly known as One-Way Chosen-Plaintext Attack (OW-CPA) secure.

OW-CPA security of Goppa-code McEliece has withstood four decades of cryptanalysis. As mentioned in the Classic McEliece submission (7), after 1978 there were 25 publications studying the one-wayness of the system and introducing increasingly sophisticated non-quantum attack algorithms:

- 1981 Clark–Cain (24), crediting Omura.
- 1988 Lee–Brickell (18).
- 1988 Leon (25).
- 1989 Krouk (26).
- 1989 Stern (19).
- 1989 Dumer (27).
- 1990 Coffey–Goodman (28).

- 1990 van Tilburg (29).
- 1991 Dumer (30).
- 1991 Coffey–Goodman–Farrell (31).
- 1993 Chabanne–Courteau (32).
- 1993 Chabaud (33).
- 1994 van Tilburg (34).
- 1994 Canteaut–Chabanne (35).
- 1998 Canteaut–Chabaud (36).
- 1998 Canteaut–Sendrier (37).
- 2008 Bernstein–Lange–Peters (5).
- 2009 Bernstein–Lange–Peters–van Tilborg (38).
- 2009 Finiasz–Sendrier (39).
- 2011 Bernstein–Lange–Peters (20).
- 2011 May–Meurer–Thomae (21).
- 2012 Becker–Joux–May–Meurer (22).
- 2013 Hamdaoui–Sendrier (40).
- 2015 May–Ozerov (41).
- 2016 Canto Torres–Sendrier (42).
- 2017 Leif Both, Alexander May (45).
- 2017 Ghazal Kachigar, Jean-Pierre Tillich (46)
- 2018 Elena Kirshanova (47).
- 2018 Leif Both, Alexander May (48).

- 2022 Thomas Debris-Alazard, Léo Ducas, Wessel P. J. van Woerden (49).
- 2022 Andre Esser, Emanuele Bellini (50)
- 2022 Andre Esser, Alexander May, Floyd Zveydinger (51).
- 2022 Andre Esser, Sergi Ramos-Calderer, Emanuele Bellini, José Ignacio Latorre, Marc Manzano (52).
- 2022 Andre Esser (53).
- 2022 Asuka Wakasugi, Mitsuru Tada (54).
- 2023 Shintaro Narisada, Kazuhide Fukushima, Shinsaku Kiyomoto (55).
- 2023 Andre Esser, Floyd Zveydinger (56).
- 2023 Qian Guo, Thomas Johansson, Vu Nguyen (57).
- 2023 Yu Li, Li-Ping Wang (58).
- 2023 Naoto Kimura, Atsushi Takayasu, Tsuyoshi Takagi (59).
- 2023 Daniel J. Bernstein, Tung Chou (60)
- 2023 Sreyosi Bhattacharyya, Palash Sarkar (61).
- 2024 Léo Ducas, Andre Esser, Simona Etinski, Elena Kirshanova (62).
- 2024 Shintaro Narisada, Shusaku Uemura, Hiroki Okada, Hiroki Furue, Yusuke Aikawa, Kazuhide Fukushima (63).

All the above mentioned papers are focusing on the most effective attack strategy known : ISD attack. McEliece CBC has shown remarkably stable security against all these attacks.

Though there are also many papers studying attacks that instead recover McEliece's private key from the public key G like Sendrier's "support splitting" algorithm (9), ISD remains the fastest attacking strategy.

Various studies about the possibility of successful effects on the Goppa code by the attack strategies successful to other codes consistently show that McEliece's system is

far beyond all known attacks.

3.3.1 CCA Vulnerability of McEliece - Sloppy Alice Attacks

In practice more than one-wayness is required. an attacker can try to exploit the specific structures inherent in the message to guess the plaintext or can try sending modified ciphertexts to learn from the reactions of the sender. To be secure against such attacks is known as INDistinguishability under adaptive Chosen Ciphertext Attack (IND-CCA 2). Though OW CPA security of Goppa-code McEliece has withstood four decades of cryptanalysis, CCA vulnerability has been found for the cryptosystem. One of this is against *Sloppy Alice Attack* formulated by Verheul, Doumen, Tilborg (12).

Sloppy Alice Attack : The Sloppy Alice attack assumes that the decoding algorithm to satisfy the *Maximum Error Property* (MEP) : that is on input of a vector v , the decoding algorithm will either return a codeword c at a distance $\leq t$ to v or it will return an error message.

The formulation is as follows.

Suppose Eve, attacker intercepts a ciphertext $v = mG' + e$, while Alice was sending it to Bob, where G' is the public generator matrix, sent by Alice to Bob. Then Eve:

1. *increases the number of errors* made by Alice to exactly t by changing a random coordinate (without any repetition in the coordinates) and sending the resulting codeword to Bob until an error message is returned. This means that the message prior to the error message has the maximum number of decodable errors.
2. *determines enough error-free coordinates:* .Having identified a message with exactly t errors, Eve randomly changes coordinates, other than those used in the previous step, and checks with Bob until an error message is returned again. This shows that the last coordinate was error-free before the change.
3. *determines the plaintext* by solving $v = mG$, using Gaussian elimination on the

error-free columns identified as above.

3.4 Niederreiter Cryptosystem

3.4.1 Introduction

The McEliece system has prompted a lot of followup work. This includes a “dual” variant of the McEliece Cryptosystem published by Harald Niederreiter in 1986 (8) improving efficiency while preserving the security. Originally this system used Generalized Reed Solomon (GRS) codes, which later turned out to be not a suitable choice as the presence of structural attack on them as shown by Sidelnikov and Shestakov (43) in 1992; then came another Niederreiter’s system with the same Goppa codes as used by McEliece. It had the same security as McEliece’s system(11). Niederreiter’s system differs from McEliece’s system in public-key structure, encryption mechanism, and decryption mechanism.

3.4.2 Outline of the Niederreiter cryptosystem

The Public key:

- Let H' be the parity check matrix of a Goppa code $\Gamma = (g, \alpha_0, \dots, \alpha_n)$ of length n and dimension $k = n - mt$.
- Let S be a random $(n - k) \times (n - k)$ invertible matrix and P , a random $n \times n$ permutation matrix.
- Define $H := SH'P \in \mathbb{F}_2^{n-k \times n}$.
- H is the public key.

The Secret key:

- The triple (H', S, P) is the secret key.

Encryption:

- Let $e \in \mathbb{F}_2^n$ be the plaintext, with $w_H(e) = t$.
- The ciphertext is defined as: $y = He$. Clearly recovering y is solving a syndrome decoding problem for random code which is NP complete.

Decryption:

- Compute $y' = S^{-1}y = H'Pe$.
- As $w_H(He) = t$, the efficient syndrome decoder for H' can obtain e' such that $H'e' = y'$.
- Extract plaintext $e = P^{-1}e'$.

So, mainly there was two modifications.

1. Instead of using a generator matrix of a random binary Goppa code, a parity check matrix has been used. This shows equivalence because one can easily compute parity check matrix in systematic from a generator matrix in systematic form and vice versa.
2. The inversion problem in McEliece CBC was to compute a uniform random input (a, e) given G and the ciphertext $Ga + e$, whereas Niederreiter's inversion problem is to compute a uniform random input e given H and the ciphertext He . Any attack recovering e from He can be used to recover a, e from $Ga + e$ for the same Goppa code. Multiply H to $Ga + e$ and we will get $HGa + He = He$. If we can compute e from He then we can subtract e from $Ga + e$ and use linear algebra to get a from G . This shows the equivalence in security. Hence, Niederreiter dual version is also OW-CPA.

But it has also CCA vulnerabilities. The Classic McEliece PKE/KEM achieves IND-CCA2.

3.5 The Classic McEliece PKE/KEM

Conversion of a OW-CPA PKE into a KEM , IND-CCA2 secure against all Random Oracle Model attacks is well known. This conversion is tight preserving the security level under two assumptions

1. PKE is deterministic
2. The PKE has no decryption failure for valid ciphertext.

Both of these assumptions are met by McEliece CBC ,hence Niederreiter dual version. Recent studies(13) show possibility of achieving similar tightness against broader class of Quantum Random Oracle Model(QROM) attacks.

The Classic McEliece KEM is designed to be IND-CCA2 secure from Niederreiter's dual version of McEliece CBC, which is OW-CPA secure PKE. KEM-DEM composition (44) produces a PKE with IND-CCA2 security. To construct KEM

- The session key as a hash of a uniform random input e is computed.
- Ciphertext is the original ciphertext plus a "confirmation": another cryptographic hash of e .
- After using the private key to compute e from a ciphertext, The ciphertext is recomputed (including the confirmation) and is checked if it matches.
- If decryption fails (i.e., if computing e fails or the recomputed ciphertext does not match), Instead of returning a KEM failure, a pseudorandom function of the ciphertext, specifically a cryptographic hash of a separate private key and the ciphertext is returned.

The intuitive arguments for the above mentioned practices discussed in the Classic McEliece submission (7) are

- A KEM construction published in a classic 2003 paper by Dent (14) features a tight proof of security against ROM attacks, assuming OW-CPA security of the

underlying PKE. This theorem relies on the first three items in the list above.

- A much more recent KEM construction by Saito, Xagawa, and Yamakawa (13) features a tight proof of security against the broader class of QROM attacks, under somewhat stronger assumptions. This theorem relies on the first, third, and fourth items.

3.5.1 The Classic McEliece Parameters:

As mentioned in the Classic McEliece submission (7), a Classic McEliece (CM) parameter set specifies the following:

- A positive integer m . This also defines a parameter $q = 2^m$.
- A positive integer n with $n \leq q$.
- A positive integer $t > 1$ with $mt < n$. This also defines a parameter $k = n - mt$.
- A monic irreducible polynomial $f(z) \in F_2[z]$ of degree m . This defines a representation $F_2[z]/f(z)$ of the field F_q .
- • A positive integer l and a cryptographic hash function H that outputs l bits

3.5.2 Design:

Key Generation: Given a set of Classic McEliece(CM) parameters, a user generates a CM key pair as follows:

1. Generate a uniform random monic irreducible polynomial $g(x) \in F_q[x]$ of degree t .
2. Select a uniform random sequence $(\alpha_1, \alpha_2, \dots, \alpha_n)$ of n distinct elements of F_q .
3. Compute the tn matrix $\tilde{H} = h_{i,j}$ over F_q , where $h_{i,j} = \alpha_j^{i-1} / g(\alpha_j)$ for $i = 1, \dots, t$ and $j = 1, \dots, n$.
4. Form an $mt \times n$ matrix \hat{H} over F_2 by replacing each entry $c_0 + c_1z + \dots + c_{m-1}z^{m-1}$ of \tilde{H} with a column of t bits c_0, c_1, \dots, c_{m-1} .

5. Reduce \hat{H} to systematic form $(I_{n-k} \mid T)$, where I_{n-k} is an $(n-k) \times (n-k)$ identity matrix. If this fails, go back to Step 1.
6. Generate a uniform random n -bit string s .

Public Key: The binary $(n-k) \times k$ matrix T such that $H = [I_{n-k} \mid T]$ is the parity check matrix of a Goppa code $\Gamma = (g, \alpha_0, \dots, \alpha_n)$ of length n and dimension $k = n - mt$.

Private Key: The tuple (s, Γ) , where s is a uniform random n bit string.

Encoding Subroutine: Inputs a weight t column vector $e \in \mathbb{F}_2^n$; public key $T \in \mathbb{F}_2^{(n-k) \times k}$, and returns a vector $c_0 \in \mathbb{F}_2^{n-k}$ as follows:

1. Define $H = [I_{n-k} \mid T]$.
2. Compute $c_0 = He \in \mathbb{F}_2^{n-k}$.

Decoding Subroutine:

1. Extend c_0 to $v = (c_0, 0, \dots, 0) \in \mathbb{F}_2^n$ by appending k zeroes.
2. Find the unique codeword c in the Goppa code Γ , from the private key (s, Γ) , at a distance $\leq t$ from v .
3. If there is no such codeword, return \perp .
4. Set $e = v + c$.
5. If $\text{wt}(e) = t$ and $c_0 = He$, return e ; else return \perp .

Encapsulation Subroutine: Generates a session key K and its ciphertext C using a cryptographic hash function H with an l -bit output:

1. Generate uniform random vector $e \in \mathbb{F}_2^n$ of weight t .
2. Use encoding subroutine on e and T to compute c_0 .
3. Compute $c_1 = H(2, e)$. Define $C = (c_0, c_1)$.
4. Compute key $K = H(1, e, C)$.

5. Output (K, C) .

Decapsulation Subroutine: Recovering the session key K from the ciphertext

C .

1. Split C as $c_0 \in \mathbb{F}_2^{n-k}$ and $c_1 \in \mathbb{F}_2^l$.
2. Set $b = 1$. Use decoding subroutine on (c_0, Γ) to compute e . If decoding returns \perp , set $e = s$ and $b = 0$.
3. Compute $c'_1 = H(2, e)$; if $c'_1 \neq c_1$, set $e = s$ and $b = 0$.
4. Compute $K = H(b, e, C)$.
5. Output session key K .

If C is a legitimate ciphertext then $C = (c_0, c_1)$ with $c_0 = He$ for some $e \in \mathbb{F}_2^n$ of weight t and $c_1 = H(2, e)$. The decoding algorithm will return e and the $c'_1 = c_1$ check will pass, thus $b = 1$ and K matches the session key computed in encapsulation.

3.5.3 Cryptanalysis

As mentioned earlier, breaking one wayness is same as solving the inversion problem of McEliece PKE. One of the best possible inversion attack strategy is ISD attacks, which is already discussed on the context of security of McEliece CBC. Other known attacks are *key recovery* and *chosen-ciphertext* attacks.

Key Recovery This is another inversion approach is to recover of the private key $\Gamma = (g, \alpha_1, \dots, \alpha_n)$. This isn't as difficult as brute force attack. Partial key exposure is possible such as recovery of Goppa points from the Goppa polynomial or vice versa (15). But the claim is due to the choice of Classic McEliece parameters the possibilities of Goppa points is too huge to implement such attacks.

Chosen-Ciphertext attack A traditional approach (Little similar to Sloppy Alice Attack) to chosen-ciphertext attacks against the McEliece system is to add two errors to a ciphertext $v = Gm + a$ which is same as adding two errors to the term e . Successful decryption is possible if and only if the weight of the resulting vector is t .

For that to happen there must be the case where one of the error reduces the weight having same coordinate in that position while the other makes the weight t again as the term $w_H(e) = t$. e can be obtained from the patterns of decryption failure.

The Classic McEliece PKE/KEM claims to be secure against such attacks because of two reasons.

1. The KEM decapsulation stage requires the ciphertext to include a hash of e as a confirmation, and the attacker has no way to compute the hash of a modified version of e without knowing e in the first place.
2. The KEM does not reveal decryption failures: the modified ciphertext will produce an unpredictable session key, whether or not the modified error vector has weight t . So finding e is no longer possible from the patterns of decryption failure.

3.6 Partial Information Attack on Classic McEliece

In 2022, Kirshanova and May reported algebraic attacks on the McEliece CBC assuming partial knowledge of the secret key (6). They proposed *polynomial time* algorithms for three cases:

1. Only $tm + 1$ Goppa points together with the public key is enough to recover the Goppa polynomial $g(x)$ in $\mathcal{O}(n^3)$ operations in \mathbb{F}_{2^m} . As, with typical McEliece parameters, $tm + 1 \sim \frac{n}{4}$, the knowledge of only a quarter of the Goppa points is enough to recover the Goppa polynomial.
2. With the knowledge of any $tm + 1$ points α_i and the Goppa polynomial, such that the submatrix of the public parity check matrix indexed by the known points has full rank, the remaining Goppa points can be recovered in polynomial time.
3. As a final refinement, the knowledge of only $t(m - 2) + 1$ Goppa points along with the Goppa polynomial, suffices to recover the full key.

This represents a potential paradigm shift in attacks on the McEliece CBC. However it is not known whether this methodology is directly applicable for attacking the Classic McEliece PKE/KEM (Public-key Encryption/Key-encapsulation Mechanism)(7).

Before proceeding to the algorithms for those cases, we shall have a look on an interesting feature of Goppa codes, which is The binary irreducible Goppa code $C(L, g)$ satisfies

$$C(L, g) = C(L, g^2).$$

. The proof is in the paper (6). From this result Kirshanova and May proposed an algorithm, with the name *TEST-GOPPA-POLYNOMIAL* to test whether a potential Goppa polynomial satisfies the equation

$$C(L, g) = \left\{ c \in \mathbb{F}_2^n : \sum_{i=1}^n \frac{c_i}{x - \alpha_i} \equiv 0 \pmod{g(x)} \right\}.$$

for all codewords in the span of some projected Goppa code. This algorithm *TEST-GOPPA-POLYNOMIAL* is used in the algorithms for the three cases. Another notation used in these algorithms is of the index set. Let $H = (\mathbf{h}_1 \dots \mathbf{h}_n) \in \mathbb{F}_2^{tm \times n}$ be a matrix with n columns $\mathbf{h}_i \in \mathbb{F}_2^{tm}$. Let $I \subset \{1, \dots, n\}$ be an index set. Then we denote by $H[I]$ the projection of H 's to the columns defined by $I = \{i_1, \dots, i_\ell\}$, i.e.,

$$H[I] = (\mathbf{h}_{i_1}, \dots, \mathbf{h}_{i_\ell}).$$

The respective algorithms for the above mentioned cases are:

Algorithm for case-1:

- Getting Public key $H^{\text{pub}} \in \mathbb{F}_2^{tm \times n}$, index set $I \subset \{1, \dots, n\}$ with $|I| > tm$, Goppa points $a_i \in \mathbb{F}_2^m$ with $i \in I$ as input, Compute $G[I] \in \mathbb{F}_2^{(\ell - \text{rank}(H)) \times \ell}$ as the right kernel of $H^{\text{pub}}[I]$.

- For some non-zero $m \in \mathbb{F}_2^{\ell - \text{rank}(H)}$ set $\mathbf{c} = mG[I]^T \in \mathbb{F}_2^\ell$.
- Construct c by appending to c zeros in all positions $\{1, \dots, n\} \setminus I$.
- Compute $f(x) = \sum_{i \in I} c_i \prod_{j \in I \setminus \{i\}} (x - a_j) \in \mathbb{F}_{2^m}[x]$ and factor $f(x)$ into irreducible factors over \mathbb{F}_{2^m} . Set $\mathcal{L} = \emptyset$
- For all irreducible degree- t factors $\tilde{g}(x)$ such that $\tilde{g}(x)^2$ divides $f(x)$ if $\text{Test-Goppa-Polynomial}(\tilde{g}(x), I, G[I]^T) = 1$, then $\mathcal{L} = \mathcal{L} \cup \{\tilde{g}(x)\}$.

Algorithm for case-2:

- **INPUT:** public key $HP^{\text{pub}} \in \mathbb{F}_2^{tm \times n}$, Goppa polynomial $g(x)$, index set I with $\ell := |I| > tm$ and $\text{rank}(HP^{\text{pub}}[I]) = tm$, Goppa points $\alpha_i \in \mathbb{F}_{2^m}$ with $i \in I \subset \{1, \dots, n\}$.
- While $I \neq \{1, \dots, n\}$ Pick $r \in \{1, \dots, n\} \setminus I$.
- Find $c \in \mathbb{F}_{2^m}^{tm}$ that solves the linear equation system $HP^{\text{pub}}[I]c = HP^{\text{pub}}[\{r\}]$.
- Compute $f(x) = \left(\sum_{i \in I} \frac{c_i}{x - \alpha_i} \right)^{-1} \pmod{g(x)}$.
- If $f(x)$ is of the form $x - \alpha_r$
- Output α_r ,
- Else Output FAIL.
- Set $I \leftarrow I \cup \{r\}$.

It is clear that using the above mentioned two algorithms together we can recover the Goppa polynomial and remaining Goppa points from only $tm + 1$ Goppa points and the public key.

Algorithm for case-3:

Let $\bar{I} = \{1, \dots, n\} \setminus I$ be the complement of I , and $I_c = I \cap \text{supp}(c)$, $\bar{I}_c = \bar{I} \cap \text{supp}(c)$.

where $\text{supp}(c)$ is the set of coordinates with non zero entries. It follows that

$$\sum_{i \in I} \frac{c_i}{x - \alpha_i} \equiv \sum_{i \in \bar{I}} \frac{c_i}{x - \alpha_i} \pmod{g^2(x)},$$

or equivalently,

$$\sum_{i \in I_c} \frac{c_i}{x - \alpha_i} \equiv \sum_{i \in \bar{I}_c} \frac{1}{x - \alpha_i} \pmod{g^2(x)}.$$

Let $h(x) = \sum_{i \in I_c} \frac{c_i}{x - \alpha_i}$. And another polynomial $p(x)$ is defined as

$$p(x) := \prod_{i \in \bar{I}_c} (x - \alpha_i) \text{ of } \deg(p(x)) = |\bar{I}_c| \leq t.$$

The algorithm is as follows

- On input public key $HP^{\text{pub}} \in \mathbb{F}_2^{tm \times n}$, Goppa polynomial $g(x)$, index set I , Goppa points $\alpha_i \in \mathbb{F}_{2^m}$ with $i \in I \subset \{1, \dots, n\}$, $c \in C(L, g)$ such that $k := |I_c| \leq t$. Derive $M \in \mathbb{F}_{2^m}^{2t \times 2k-1}$ and $b \in \mathbb{F}_{2^m}^{2t}$ such that $My = b$ over \mathbb{F}_{2^m} .
- If $My = b$ has a unique solution $y = (p_0, \dots, p_{k-1}, p'_0, \dots, p'_{k-2}) \in \mathbb{F}_{2^{2k-1}}$
- Then $p(x) \leftarrow x^k + \sum_{i=0}^{k-1} p_i x^i \in \mathbb{F}_{2^m}[x]$
- Factor $p(x)$ over $\mathbb{F}_{2^m}[x]$ into linear factors $p(x) = \prod_{i=1}^k (x - \alpha_i)$.
- Return $A_{I_c} \leftarrow \{\alpha'_1, \dots, \alpha'_k\}$.
- Else Output FAIL.

For detailed explanations, proofs and results of these algorithms (6) is referred.

4 | BIKE

4.1 Introduction

BIKE (72) is a Key Encapsulation Mechanism (KEM) that uses Quasi-Cyclic Moderate Density Parity-Check (QC-MDPC) codes that is one of the fourth round contenders in NIST PQC. BIKE is based on the Niederreiter framework, hence encoding is done with the parity check matrix in its systematic format. Using Niederreiter framework instead of analogous McEliece framework reduces the communication bandwidth to half with the trade-off of costlier polynomial inversion for key generation.

After the second round in NIST the proposal for BIKE included six variants: BIKE-1, BIKE-2, BIKE-3, BIKE-1-CCA, BIKE-2-CCA, and BIKE-3-CCA. Following NIST's recommendation to simplify the proposal, the BIKE designers consolidated it into a single version, BIKE-2-CCA, now simply called BIKE. This version is different from the previously proposed BIKE-2-CCA (after round 2) and has three parameter sets (r, w, t) , providing security for three levels :- Level-1 ,Level-3, Level-5. BIKE was initially proposed for ephemeral key use (using a public/private key pair only once.) giving forward secrecy. For such usage achieving IND-CPA is sufficient. It has now been claimed to also support static key use (using same keys indefinitely) which require IND-CCA security. The IND-CPA security of BIKE is based upon the well studied quasi-cyclic variants of computationally hard code-based problems: the decisional Quasi-cyclic Syndrome Decoding (QCSD) and the decisional Quasi-cyclic Codeword Finding (QCCF) problems. BIKE is claimed to have IND-CCA security with the adaptation of Fujisaki-Okamoto transformation (73)(from δ - correct PKE to

an IND-CCA KEM) without compromising the performance. The decoder associated with BIKE is Black-Gray-Flip (BGF) (74) with very low decryption failure rate(DFR).

The minimum memory requirements for BIKE:

Quantity	Size	Level 1	Level 3	Level 5
Private key	$\ell + w \cdot \lceil \log_2(r) \rceil$	2,244	3,346	4,640
Public key	r	12,323	24,659	40,973
Ciphertext	$r + \ell$	12,579	24,915	41,229

Table 4.1: Size of Keys and Ciphertext for Different Security Levels

4.2 Instance of Niederreiter scheme with QC-MDPC

The McEliece scheme can be implemented with QC-MDPC codes. A public key encryption scheme with QC-MDPC codes in the Niederreiter template is discussed below.

Private Key: $(h_0, h_1) \in \mathcal{H}_w$

Public Key: $h = h_1 h_0^{-1} \in \mathcal{R}$

Encryption: $(e_0, e_1) \in \mathcal{E}_t \mapsto s = e_0 + e_1 h \in \mathcal{R}$

Decryption: $s \in \mathcal{R} \mapsto \text{decoder}(s h_0, h_0, h_1) \in \mathcal{E}_t$

This system is based on the hardness of the following two problems for security of:

Key: distinguish $h_1 h_0^{-1}$ from random, $(h_0, h_1) \xleftarrow{\$} \mathcal{H}_w$

Message: distinguish $(e_0 + e_1 h, h)$ from random, $((e_0, e_1), h) \xleftarrow{\$} \mathcal{E}_t \times \mathcal{R}$

4.3 Overview of BIKE

4.3.1 Specification

1. **System Parameters:** Based on the required level of security (denoted by λ), system parameters r, w, l , and t are determined.

- r is the length of circulant blocks (equivalent to p as discussed earlier).

Given $r = N/n_0$ and $n_0 = 2$ in BIKE, we have $N = 2r$. Moreover, since $r = N - K$, we have $N = 2r$ or $r = K$. It should be sufficiently large to result in (together with w and t) a low level of DFR such that the required security level λ is finally met (in BIKE, three security levels 1, 3, and 5 have been considered, corresponding to the security of AES-128, AES-192, and AES-256, respectively. To satisfy each level, a separate set of system parameters are recommended).

- w is the row weight of the parity check matrix which is an even positive integer (i.e., $w/2$ is odd).
- t is the Hamming weight of the error vector which is a positive integer.
- l is the size of the generated (shared) symmetric key which is a positive integer.

Suggested parameters and DFR for BIKE for different security level are :

Table 4.2: Security Parameters and DFR

Security Level	r	w	t	DFR
Level 1	12,323	142	134	2^{-128}
Level 3	24,659	206	199	2^{-192}
Level 5	40,973	274	264	2^{-256}

The DFR values are based on BGF decoder and BGF decoder parameters (Discussed later in this chapter) for three security levels namely level 1,3 and 5 , corresponding to the security of AES-128, AES-192, and AES-256, respectively are

- **For Level 1:**
 - Number of Iterations (NbIter): 5
 - τ : 3
 - $\text{threshold}(S, i) = \max(\lfloor 0.0069722 \cdot S + 13.530 \rfloor, 36)$

- **For Level 3:**

- Number of Iterations (NbIter): 5
- τ : 3
- $\text{threshold}(S, i) = \max(\lfloor 0.005265 \cdot S + 15.2588 \rfloor, 52)$

- **For Level 5:**

- Number of Iterations (NbIter): 5
- τ : 3
- $\text{threshold}(S, i) = \max(\lfloor 0.00402312 \cdot S + 17.8785 \rfloor, 69)$

2. **Hash Functions:** In BIKE, three hash functions \mathcal{H} , \mathcal{L} , and \mathcal{K} are uniformly selected at random that are modeled as random oracles. \mathcal{H} takes an l -bit sequence and generates a $2r$ -bit sequence of Hamming weight t (i.e., $\mathcal{H} : \{0, 1\}^l \rightarrow \{0, 1\}^{2r}$). It is generated by the algorithm WSHAKE256-PRF(seed, len, wt) mentioned in the context of pseudorandom bit generation. Similarly for \mathcal{L} and \mathcal{K} we have $\mathcal{L} : \{0, 1\}^r \rightarrow \{0, 1\}^r$ and $\mathcal{K} : \{0, 1\}^{2r} \rightarrow \{0, 1\}^l$, respectively.

Select a decoder decoder, which takes as input $s \in \mathcal{R}$ and $(h_0, h_1) \in H_w$. The call $\text{decoder}(s, h_0, h_1)$ returns either $(e_0, e_1) \in \mathcal{R}^2$ such that $e_0 h_0 + e_1 h_1 = s$, or the failure symbol \perp . The decoding failure rate is defined as:

$$DFR(\text{decoder}) = \Pr[(e_0, e_1) \neq \text{decoder}(e_0 h_0 + e_1 h_1, h_0, h_1)]$$

when $((h_0, h_1), (e_0, e_1))$ is drawn uniformly from $H_w \times E_t$.

4.3.2 BIKE KEM

Key Generation:

- *Private Key:* Generate $h_0, h_1 \leftarrow R$ both of weight $|h_0| = |h_1| = w/2$, where R is a cyclic polynomial ring $F_2[X]/(X^r - 1)$ (equivalently, h_0 and h_1 can be

considered as $r \times 1$ column vectors). Then, select σ at random (uniformly) from the message space $M = 0, 1^l$. Finally, set the private key as $sk = (h_0, h_1, \sigma)$.

- *Public Key*: : Compute $h = h_1 \cdot h_0^{-1}$ and send it to the other party as the public key pk .

Encapsulation subroutine: $h \mapsto K, c$

Input: $h \in \mathcal{R}$

Output: $K \in \mathcal{K}, c \in \mathcal{R} \times \mathcal{M}$

1: $m \xleftarrow{R} \mathcal{M}$

2: $(e_0, e_1) \leftarrow \mathcal{H}(m)$

3: $c \leftarrow (e_0 + e_1 h, m \oplus \mathcal{L}(e_0, e_1))$

4: $K \leftarrow \mathcal{K}(m, c)$

Decapsulation subroutine: $(h_0, h_1, \sigma), c \mapsto K$

Input: $((h_0, h_1, \sigma) \in \mathcal{H}_w \times \mathcal{M}, c = (c_0, c_1) \in \mathcal{R} \times \mathcal{M})$

Output: $K \in \mathcal{K}$

1: $e' \leftarrow \text{decoder}(c_0 h_0, h_0, h_1)$

2: $m' \leftarrow c_1 \oplus \mathcal{L}(e')$

3: **if** $e' = \mathcal{H}(m')$ **then** $K \leftarrow \mathcal{K}(m', c)$ **else** $K \leftarrow \mathcal{K}(\sigma, c)$

with the convention $\perp = (0, 0)$

4.3.3 Decoder

Black-Gray-Flip (BGF) decoder has been used in BIKE. The BGF decoder is a variant of the Black-Gray (BG) decoder, which is a complex version of the BF decoder. The main difference between BG and the original BF decoder is that a BG decoder may flip back the value of a bit (variable) node (or an error bit, equivalently) if it is convinced that the bit was mistakenly flipped in the previous iteration. A simple version is the Time-to-Live (TTL) mechanism. However, in BG, a more complex version of TTL is used. The decoder creates two lists of bit nodes (called Black and

Gray lists) to keep track of the *uncertain* bit nodes to be flipped. The *Black* list keeps track of the flipped bit nodes while the *Gray* list keeps track of the bit nodes with unsatisfied parity checks close to the threshold without exceeding it.

In every iteration of the BG decoder, three steps are performed:

1. The decoder decides if a bit node should be flipped by comparing unsatisfied parity checks with a threshold b (similar to the BF decoder). It computes black and gray lists.
2. Another BF iteration considers only black list bit nodes, updating values if unsatisfied parity checks exceed the threshold $\left\lceil \frac{d+1}{2} \right\rceil + 1$ (where $d = \frac{w}{2}$).
3. Similarly, a BF iteration for gray list bit nodes is performed.

If the syndrome is zero, the error vector is returned; otherwise, iterations continue until the maximum is reached.

The BGF decoder, used in BIKE, starts with one BG iteration followed by several BF iterations until the syndrome is zero or the maximum iterations are reached. BGF requires fewer steps than BG to achieve lower DFR. For NI iterations, BG needs $3 \times NI$ steps, whereas BGF needs $3 \times (NI - 1)$.

The BGF decoder is the most efficient BF variant in terms of DFR and complexity. In BIKE, thresholds for BGF are derived from extensive simulations and are crucial system parameters, targeting DFRs of 2^{-128} , 2^{-192} , and 2^{-256} for respective security levels.

Inputs to the BGF decoder are: a vector $s \in \mathbb{F}_2^r$ and a matrix $H \in \mathbb{F}_2^{r \times n}$. The matrix H is of the form $[H_0 \mid H_1]$, where H_0 and H_1 are circulant blocks derived from $(h_0, h_1) \in H_w$. The algorithm is defined for every parameter sets (r, w, t) and uses additional parameters

1. NbIter: the number of iterations to be performed.
2. τ : a threshold gap used to determined the size of the 'gray' set of positions

3. $\text{Threshold}(S, i)$: This is the threshold function and used as a threshold selection rule .It depends on the syndrome weight S , the iteration number i , and various system parameters.
4. $\text{ctr}(H, s, j)$. This function computes a quantity referred to as the counter (also known as the number of unsatisfied parity-checks) of j . It is the number of '1' (set bits) that appear in the same position in the syndrome s and in the j -th column of the matrix H .

The BGF algorithm is the following.

Algorithm 1 Black-Gray-Flip (BGF)

- **Parameters:** $r, w, t, d = w/2, n = 2r; \text{NbIter}, \tau, \text{threshold}$ (see text for details)
 - **Require:** $s \in \mathbb{F}_r, H \in \mathbb{F}_2^{n \times k}$
 - $e \leftarrow 0^n$
 - **for** $i = 1, \dots, \text{NbIter}$ **do**
 - $T \leftarrow \text{threshold}(s + eH^T, i)$
 - $e_{\text{black}}, e_{\text{gray}} \leftarrow \text{BFilter}(s + eH^T, e, T, H)$
 - **if** $i = 1$ **then**
 - * $e' \leftarrow e_{\text{black}}$
 - **else**
 - * $e' \leftarrow \text{BFMaskedIter}(s + eH^T, e_{\text{black}}, (d + 1)/2 + 1, H)$
 - * $e'' \leftarrow \text{BFMaskedIter}(s + eH^T, e_{\text{gray}}, (d + 1)/2 + 1, H)$
 - **if** $s = eH^T$ **then**
 - * **return** e
 - **else**
 - * **return** \perp
 - **end for**
 - **return** e
 - **procedure** $\text{BFilter}(s, e, T, H)$
 - **for** $j = 0, \dots, n - 1$ **do**
 - * **if** $\text{ctr}(H_{*,j}s) \geq T$ **then**
 - $e_j \leftarrow e_j + 1$
 - **if** $\text{ctr}(H_{*,j}s) \geq T - \tau$ **then** $\text{gray} \leftarrow \text{gray} + 1$
 - **else** $\text{black} \leftarrow \text{black} + 1$
 - **return** $e_{\text{black}}, e_{\text{gray}}$
 - **procedure** $\text{BFMaskedIter}(s, e, \text{mask}, T, H)$
 - **for** $j = 0, \dots, n - 1$ **do**
 - * **if** $\text{ctr}(H_{*,j}s) \geq T$ **then**
 - $e_j \leftarrow e_j \oplus \text{mask}_j$
 - **return** e
-

4.3.4 Pseudo-random Bits Generation

Key Generation, Encapsulation, and Decapsulation of BIKE KEM involve pseudorandom bits stream generation. The algorithms are based on *SHAKE256*. With an input of 256 bits seed *SHAKE256* outputs μ blocks each of 1088 bits. The output for that seed is denoted by $SHAKE256(\text{seed}, \mu)$, the least significant ν bits of $SHAKE256(\text{seed}, \text{ceil}(\nu/1344))$ are denoted by $SHAKE256\text{-Stream}(\text{seed}, \nu)$.

The pseudo-random bit generation algorithm with no constraints on the output is **GenPseudoRand(seed, len)** : It takes 256-bit seed as input and outputs $SHAKE256\text{-Stream}(\text{seed}, \text{len})$. The pseudo-random bit generation algorithm with a specific weight w output is **WSHAKE256-PRF(seed, len, wt)** which on *input*: seed (32 bytes), len, wt *outputs* a list (wlist) of wt distinct elements in $0, \dots, \text{len} - 1$.

- : wlist \leftarrow () [() is an empty list]
 - $s_0, \dots, s_{wt-1} \leftarrow SHAKE256\text{-Stream}(\text{seed}, 32 \cdot wt)$
 - For $i = (wt - 1), \dots, 0$ do (i decreasing from $wt - 1$ to 0):
 - $\text{pos} \leftarrow i + b(\text{len} - i) \cdot \frac{s_i}{2}$
 - $\text{wlist} \leftarrow \text{wlist}, (\text{pos} \in \text{wlist}) ? i : \text{pos}$
6. Return wlist

This algorithm is used in the key generation.

4.4 Security of BIKE

The proof of IND-CPA security of the BIKE depends on the difficulty of solving the decisional Quasi-cyclic Syndrome Decoding (QCSD) and the decisional Quasi-cyclic Codeword Finding (QCCF) problems (discussed in the chapter preliminaries). The best known algorithms for solving these problems are information set decoding (ISD) and its variants discussed in the chapter- *Classic McEliece*. The work factor for solving

linear decoding problems using ISD was shown to be asymptotically equivalent across all variants of ISD.

The system parameters for BIKE are selected according to the following guidelines:

- **BIKE Message Security:**

$$\text{WF}(\text{QCSD}_\rho) \approx \frac{\text{WF}_{\text{ISD}}(2r, r, t)}{\sqrt{r}}$$

- **BIKE Key Security:**

$$\text{WF}(\text{QCCF}_{\rho,w}) \approx \frac{\text{WF}_{\text{ISD}}(2r, r, w)}{r}$$

where $\text{WF}(\text{QCSD}_\rho)$ and $\text{WF}(\text{QCCF}_{\rho,w})$ denote the average cost for finding a witness for Problems QCSD and QCCF (mentioned in preliminaries), respectively and WF_{ISD} is the average cost of the best known ISD variant for the generic decoding of linear codes.

For security against CCA, another factor is crucial:- correctness of the decoder. As discussed earlier, the decoder used in BIKE (in the decoding step of decapsulation) is BGF decoder, which has a non-zero probability of decoding failure. The decoder is called δ correct if probability of decoding failure is at most δ over all message and key space. With non-negligible the scheme can be prone to reaction attacks. With BGF in use the current version of BIKE has been claimed to be secure against CCA by the BIKE team.

4.4.1 OW-CPA Security of BIKE:

BIKE is one way secure against chosen plaintext attack(OW-CPA). To prove the one way security of BIKE, it is enough to show the generalised OW-CPA security of the PKE instance of Niederreiter like scheme with QC-MDPC(discussed earlier in this

chapter[4.2]). This security proof is collected the the BIKE (72) submission for the 4th round.

Game G_3 (OW-CPA)	Game G_4	$D(h)$
1: $(h_0, h_1) \xleftarrow{\$} \mathcal{H}_w$	1: $(h_0, h_1) \xleftarrow{\$} \mathcal{H}_w$	1:
2: $h \leftarrow h_1 h_0^{-1}$	2: $h \xleftarrow{\$} \mathcal{R}_{\text{odd}}$	2:
3: $(e_0^*, e_1^*) \xleftarrow{\$} \mathcal{E}_t$	3: $(e_0, e_1^*) \xleftarrow{\$} \mathcal{E}_t$	3: $(e_0^*, e_1^*) \xleftarrow{\$} \mathcal{E}_t$
4: $s^* \leftarrow e_0^* + e_1^* h$	4: $s^* \leftarrow e_0 + e_1^* h$	4: $s^* \leftarrow e_0^* + e_1^* h$
5: $e \leftarrow A'(h, s^*)$	5: $e \leftarrow A'(h, s^*)$	5: $e \leftarrow A'(h, s^*)$
6: return QCSD(e, h, s^*)	6: return QCSD(e, h, s^*)	6: return QCSD(e, h, s^*)

By the attack game we are going to show that If there exist an OW-CPA adversary A' for the PKE , then there exist a distinguisher D against QCCF (quasi cyclic codeword finding problem) such that

$$Adv_{\text{OW-CPA}}^{\text{PKE}}(A') \leq Adv_{\text{IND-QCCF}}^{\text{PKE}}(D) + Adv_{\text{OW-QCSD}}^{\text{PKE}}(A''_0) \text{ Proof.}$$

1. The difference between G_3 and G_4 lies solely on the way h is selected. The distinguisher D defined in Table 11 verifies

$$Adv^{G_3}(A') = \Pr \left[D(h, h_2^1) \notin (h_0, h_1) \in \mathcal{H}_2 \right]$$

$$Adv^{G_4}(A') = \Pr \left[D(h, h_2^1) \mid h \leftarrow \mathcal{R}_{\text{bad}} \right]$$

and thus

$$\left| Adv^{G_3}(A') - Adv^{G_4}(A') \right| = Adv_{\text{IND}}^{\mathcal{H}_2}(D).$$

2. The adversary A' can be viewed as a decoder against QSD. It verifies

$$Adv^{G_4}(A') = Adv_{\text{QSD}}^{G_4}(A').$$

Finally, since G_3 is the OW-CPA game against PKE,

$$Adv_{\text{OW-CPA}}^{\mathcal{R}_{\text{bad}}}(A') \leq Adv_{\text{IND-QSD}}^{\mathcal{H}_2}(D) + Adv_{\text{OW}}^{G_4}(A').$$

This proves the one wayness of BIKE.

5 | Conclusion

Both Classic McEliece and BIKE represent robust approaches to post-quantum cryptography, each with its strengths and potential applications. Classic McEliece offers a mature and well-understood solution with a long history of security analysis, while BIKE introduces innovative techniques that promise efficient implementation and strong theoretical security. The choice between these schemes would depend on specific deployment requirements, such as performance constraints and risk tolerance, highlighting the diversity and depth of the NIST PQC standardization process.. A parameter set for a proposal is said to match the security level of category one, three or five if the scheme instantiated with the corresponding parameters is at least as hard to break as AES-128, AES-192 or AES-256 respectively. Classic McEliece presents one parameter set for each level 1 and level 3 security and three different parameter sets for level 5. BIKE has one parameter set for each security level. Here is some comparison of these two with respect to cryptographic parameters size, performance efficiencies and bit complexity estimates for ISD attack and its variant attacks on all the parameter sets submitted, collected from the paper *Syndrome Decoding Estimator* published by Andre Esser and Emanuele Bellini (50).

Table 5.1: Cryptographic parameters of the Classic McEliece algorithm

NIST level	Designation	Public key size (bytes)	Private key size (bytes)	Ciphertext size (bytes)	Session key size (bytes)
L1	mceliece348864	261120	6492	96	32
L3	mceliece460896	524160	13608	156	32
L5a	mceliece6688128	1044992	13932	208	32
L5b	mceliece6960119	1047319	13948	194	32
L5c	mceliece8192128	1357824	14160	256	32

As for performance efficiency in terms of computational costs, Classic McEliece required significantly more computational resources for a key generation across all

Table 5.2: Performance indicators of the Classic McEliece algorithm (AVX512), cycles

NIST level	Designation	KeyGen	Encaps	Decaps
L1	mceliece348864	56705880	36457	127140
L3	mceliece460896	153626194	76086	263046
L5a	mceliece6688128	443476968	171442	306212
L5b	mceliece6960119	316995472	144678	286596
L5c	mceliece8192128	486159290	156945	310097

Table 5.3: Cryptographic and performance metrics of the BIKE algorithm

NIST Level	Public key (bits)	Private key (bits)	Ciphertext (bits)	Performance Indicators (AVX512-enabled) (cycles)		
				KeyGen	Encaps	Decaps
L1	12323	2244	12579	589	97	132
L3	24659	3346	24915	1823	223	387
L5	40973	4640	41229	3659	400	822

security levels. This aspect might limit its adoption in environments where computational power is a primary concern, despite its efficiency in the encapsulation and decapsulation processes. Meanwhile, BIKE demonstrated an overall balanced performance profile with relative efficiencies across all procedures.

The complexities listed are the best complexities obtained when considering the logarithmic and cube-root memory access model. First table shows the complexities Classic McEliece parameter sets and the next one shows for BIKE parameter sets.

Table 5.4: bit estimates for Classic McEliece

	Category 1 (n = 3488)		Category 3 (n = 4608)		Category 5 (n = 6688)		Category 5 (n = 6960)		Category 5 (n = 8192)	
	T	M	T	M	T	M	T	M	T	M
Prange	173	22	217	23	296	24	297	24	334	24
Stern	151	50	193	60	268	80	268	90	303	109
Both-May	143	88	182	101	250	136	249	137	281	141
May-Ozerov	141	89	180	113	246	146	246	140	276	144
B.JMM	142	97	183	121	248	160	248	163	278	189

It should be noted that the parameter set intended for level-3 security in the Classic McEliece submission falls short of the desired security level after relevant analysis. Meanwhile, BIKE continues to show promise in meeting the targeted security levels. Both being a tough contender in the 4th round, Classic McEliece and BIKE has different advantages in different scenarios. In scenarios where computational resources and storage are not stringent, Classic McEliece might be an appropriate choice due to its relative performance efficiency. Conversely, in situations with strict

Table 5.5: Bit estimates for BIKE

	Category 1 ($n = 24646$)		Category 3 ($n = 49318$)		Category 5 ($n = 81946$)	
	T	M	T	M	T	M
Prange	167	28	235	30	301	32
Stern	146	40	211	43	277	45
Both-May	147	38	212	41	276	63
May-Ozerov	146	55	211	57	276	61
BJMM	147	38	211	59	277	63

size limitations, BIKE might be the more suitable algorithm.

Bibliography

- [1] Robert J. McEliece. "A public-key cryptosystem based on algebraic coding theory", DSN Progress Report 42-44, pp. 114-116. (1978).
- [2] Berlekamp, E.R., McEliece, R.J. and van Tilborg, H., "On the Inherent Intractability of Certain Coding Problems," IEEE Trans. Inform. Theory, IT-24, (1978).
- [3] Goppa, Valery Denisovich, "A new class of linear error-correcting codes", Probl. Inf. Transm. 6: 300-304, (1970).
- [4] Eugene Prange. "The use of information sets in decoding cyclic codes." IRE Transactions on Information Theory (1962).
- [5] D. J. Bernstein, T. Lange, and C. Peters. "Attacking and defending the McEliece cryptosystem". In J. A. Buchmann and J. Ding, editors, PostQuantum Cryptography, Second International Workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17-19, (2008).
- [6] Elena Kirshanova, Alexander May. "Breaking Goppa-based McEliece with hints." SCN (2022); Information and Computation, vol. 293, (2023).
[<https://eprint.iacr.org/2022/525>]
- [7] Daniel J Bernstein *et al.* , "Classic McEliece: conservative code-based cryptography".
[[https://https://classic.mceliece.org/nist/mceliece-submission-20221023.pdf](https://classic.mceliece.org/nist/mceliece-submission-20221023.pdf)]
(2022).

- [8] Harald Niederreiter. "Knapsack-type cryptosystems and algebraic coding theory." *Problems of Control and Information Theory*. 1986.
- [9] Nicolas Sendrier. "Finding the permutation between equivalent linear codes: The support splitting algorithm." *IEEE Transactions on Information Theory*. 2000.
- [10] V. M. Sidelnikov and S. O. Shestakov. On insecurity of cryptosystems based on generalized ReedSolomon codes. *Discrete Mathematics and Applications*, 1992.
- [11] Li Yuanxing, Robert H. Deng, and Xinmei Wang. On the equivalence of mceliece's and niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*,:271-273, 1994.
- [12] Eric R. Verheul, Jeroen M. Doumen, Henk C. A. van Tilborg. "Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece public-key cryptosystem." *Information, coding and mathematics*. 2002.
- [13] Tsunekazu Saito, Keita Xagawa, Takashi Yamakawa. "Tightly-secure key-encapsulation mechanism in the quantum random oracle model." *Eurocrypt* 2018. <https://eprint.iacr.org/2017/1005>
- [14] Alexander W. Dent. "A designer's guide to KEMs." Cirencester 2003. [<https://eprint.iacr.org/200>]
- [15] Pierre Loidreau and Nicolas Sendrier. Weak keys in the McEliece public-key cryptosystem. *IEEE Trans. Information Theory*, 47(3):1207–1211, 2001.
- [16] Goppa, Valery Denisovich, "A new class of linear error-correcting codes", *Probl. Inf. Transm.* 6: 300–304, (1970)
- [17] Nicholas J. Patterson, The algebraic decoding of Goppa codes, *IEEE Transactions on Information Theory* 21 (1975), 203–207. DOI: 10.1109/ TIT.1975.1055350.
- [18] Lee, P.J., Brickell, E.F. (1988). An Observation on the Security of McEliece's Public-Key Cryptosystem. In: Barstow, D., et al. *Advances in Cryptology* —

- EUROCRYPT '88. EUROCRYPT 1988. Lecture Notes in Computer Science, vol 330. Springer, Berlin, Heidelberg.
- [19] Jacques Stern. A method for finding codewords of small weight. In G´erard D. Cohen and Jacques Wolfmann, editors, Coding theory and applications, volume 388 of Lecture Notes in Computer Science, pages 106–113. Springer, New York, 1989.
- [20] Daniel J. Bernstein, Tanja Lange, Christiane Peters. "Smaller decoding exponents: ball-collision decoding." Crypto 2011. <https://eprint.iacr.org/2010/585>
- [21] Alexander May, Alexander Meurer, Enrico Thomae. "Decoding random linear codes in $\mathcal{O}(2^{0.054n})$." Asiacrypt 2011. <https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/paper/decoding.pdf>
- [22] Anja Becker, Antoine Joux, Alexander May, Alexander Meurer. "Decoding random binary linear codes in $2^{n/20}$: How $1+1=0$ improves information set decoding." Eurocrypt 2012. <https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/paper/isd-extended.pdf>
- [23] Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In EUROCRYPT'2010, pages 235–256, 2010.
- [24] George C. Clark, Jr., and J. Bibb Cain. "Error-correcting coding for digital communication." 1981. Credits Omura for an ISD attack.
- [25] Jeffrey S. Leon. "A probabilistic algorithm for computing minimum weights of large error-correcting codes." IEEE Transactions on Information Theory. 1988.
- [26] Evgueni A. Krouk. "Decoding complexity bound for linear block codes." Problemy Peredachi Informatsii. 1989. <http://www.mathnet.ru/eng/ppi665>
- [27] Ilya I. Dumer. "Two decoding algorithms for linear codes." Problemy Peredachi Informatsii. 1989. <http://www.mathnet.ru/eng/ppi635>

- [28] John T. Coffey, Rodney M. Goodman. "The complexity of information set decoding." IEEE Transactions on Information Theory. 1990.
- [29] Johan van Tilburg. "On the McEliece public-key cryptosystem." Crypto 1988, published 1990
- [30] Ilya I. Dumer. "On minimum distance decoding of linear codes." Fifth joint Soviet-Swedish international workshop on information theory. 1991.
- [31] John T. Coffey, Rodney M. Goodman, P. Farrell. "New approaches to reduced complexity decoding." Discrete and Applied Mathematics. 1991.
- [32] Herve Chabanne, Bernard Courteau. "Application de la méthode de décodage itérative d'Omura à la cryptanalyse du système de McEliece." 1993.
- [33] Florent Chabaud. "Asymptotic analysis of probabilistic algorithms for finding short codewords." EUROCODE 1992, published 1993.
- [34] Johan van Tilburg. "Security-analysis of a class of cryptosystems based on linear error-correcting codes." PhD thesis, Technische Universiteit Eindhoven. 1994.
- [35] Anne Canteuat, Herve Chabanne. "A further improvement of the work factor in an attempt at breaking McEliece's cryptosystem." EUROCODE 1994.
<https://hal.inria.fr/inria-00074443>
- [36] Anne Canteaut, Florent Chabaud. "A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511." IEEE Transactions on Information Theory. 1998. https://www.rocq.inria.fr/secret/Anne.Canteaut/Publications/Canteaut_Chabaud98.pdf
- [37] Anne Canteaut, Nicolas Sendrier. "Cryptanalysis of the original McEliece cryptosystem." Asiacrypt 1998. https://www.rocq.inria.fr/secret/Anne.Canteaut/Publications/Canteaut_Sendrier98.pdf

- [38] Daniel J. Bernstein, Tanja Lange, Christiane Peters, Henk C. A. van Tilborg. "Explicit bounds for generic decoding algorithms for code-based cryptography." WCC 2009.
- [39] Matthieu Finiasz, Nicolas Sendrier. "Security bounds for the design of code-based cryptosystems." Asiacrypt 2009.
<https://eprint.iacr.org/2009/414>
- [40] Yann Hamdaoui, Nicolas Sendrier. "A non asymptotic analysis of information set decoding." 2013. <https://eprint.iacr.org/2013/162>
- [41] Alexander May, Ilya Ozerov. "On computing nearest neighbors with applications to decoding of binary linear codes." Eurocrypt 2015. <https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/paper/codes.pdf>
- [42] Rodolfo Canto Torres, Nicolas Sendrier. "Analysis of information set decoding for a sub-linear error weight." PQCrypto 2016.
<https://hal.inria.fr/hal-01244886v1/document>
- [43] Vladimir M. Sidelnikov, Sergey O. Shestakov. "On insecurity of cryptosystems based on generalized Reed-Solomon codes." Discrete Mathematics and Applications. 1992.
- [44] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput., 33(1):167–226, January 2004.
- [45] Leif Both, Alexander May. "Optimizing BJMM with nearest neighbors: full decoding in $2^{2n/21}$ and McEliece security." WCC 2017.
- [46] Ghazal Kachigar, Jean-Pierre Tillich. "Quantum information set decoding algorithms." PQCrypto 2017.
- [47] Elena Kirshanova. "Improved quantum information set decoding." PQCrypto 2018.

- [48] Leif Both, Alexander May. "Decoding linear codes with high error rate and its impact for LPN security." PQCrypto 2018.
- [49] Thomas Debris-Alazard, Léo Ducas, Wessel P. J. van Woerden. "An algorithmic reduction theory for binary codes: LLL and more." IEEE Transactions on Information Theory. 2022.
- [50] Andre Esser, Emanuele Bellini. "Syndrome decoding estimator." PKC 2022.
- [51] Andre Esser, Alexander May, Floyd Zweydinger. "McEliece needs a break—solving McEliece-1284 and Quasi-Cyclic-2918 with modern ISD." EUROCRYPT 2022.
- [52] Andre Esser, Sergi Ramos-Calderer, Emanuele Bellini, José Ignacio Latorre, Marc Manzano. "Hybrid decoding – classical-quantum trade-offs for information set decoding." PQCrypto 2022.
- [53] Andre Esser. "Revisiting nearest-neighbor-based information set decoding." 2022.
- [54] Asuka Wakasugi, Mitsuru Tada. "Security analysis for BIKE, Classic McEliece and HQC against the quantum ISD algorithms." 2022.
- [55] Shintaro Narisada, Kazuhide Fukushima, Shinsaku Kiyomoto. "Multiparallel MMT: faster ISD algorithm solving high-dimensional syndrome decoding problem." IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. 2023.
- [56] Andre Esser, Floyd Zweydinger. "New time-memory trade-offs for subset sum – Improving ISD in theory and practice." Eurocrypt 2023.
- [57] Qian Guo, Thomas Johansson, Vu Nguyen. "A new sieving-style information-set decoding algorithm." 2023.

- [58] Yu Li, Li-Ping Wang. "Security analysis of the Classic McEliece, HQC and BIKE schemes in low memory." *Journal of Information Security and Applications*. 2023. <https://eprint.iacr.org/2023/428>
- [59] Naoto Kimura, Atsushi Takayasu, Tsuyoshi Takagi. "Memory-efficient quantum information set decoding algorithm." *ACISP 2023*.
- [60] Daniel J. Bernstein, Tung Chou. "CryptAttackTester: formalizing attack analyses." 2023.
- [61] Sreyosi Bhattacharyya, Palash Sarkar. "Concrete time/memory trade-offs in generalised Stern's ISD algorithm." *Indocrypt 2023*.
- [62] Léo Ducas, Andre Esser, Simona Etinski, Elena Kirshanova. "Asymptotics and improvements of sieving for codes." *Eurocrypt 2024*.
- [63] Shintaro Narisada, Shusaku Uemura, Hiroki Okada, Hiroki Furue, Yusuke Aikawa, Kazuhide Fukushima. "Revisiting the May–Meurer–Thomae algorithm — solving McEliece-1409 in one day." 2024.
- [64] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, p. 120–126, Feb. 1978.
- [65] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, pp. 203–209, Jan. 1987.
- [66] A. K. Lenstra and H. W. Lenstra, *The development of the number field sieve*, vol. 1554 of *Lecture notes in mathematics*. Berlin [etc.]: Springer-Verlag, 1993.
- [67] Grover L.K.: From Schrödinger's equation to quantum search algorithm, *American Journal of Physics*, 69(7): 769–777, 2001. Pedagogical review of the algorithm and its history.

- [68] Shor, Peter W. (October 1997). "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". *SIAM Journal on Computing*. 26 (5): 1484–1509.
- [69] Bos2017CRYSTALS, CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM, Joppe W. Bos and Léo Ducas and Eike Kiltz and Tancrede Lepoint and Vadim Lyubashevsky and John M. Schanck and Peter Schwabe and Damien Stehlé, 2018 IEEE European Symposium on Security and Privacy (EuroS&P),2017,pages: 353-367,
<https://api.semanticscholar.org/CorpusID:20449721>
- [70] CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme, Léo Ducas and Eike Kiltz and Tancrede Lepoint and Vadim Lyubashevsky and Peter Schwabe and Gregor Seiler and Damien Stehlé, *Journal of ACM*,2018, volume-2018, pages: 238-268,
<https://api.semanticscholar.org/CorpusID:3593118>
- [71] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and Jurjen Bos. HQC. Technical report, National Institute of Standards and Technology, 2020.
- [72] BIKE—Bit Flipping Key Encapsulation. <https://bikesuite.org/>
- [73] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *J. Cryptology*, volume 26, pages 80–101, 2013.
- [74] Nir Drucker, Shay Gueron, and Dusan Kostic. QC-MDPC decoders with several shades of gray. In Jintai Ding and Jean-Pierre Tillich, editors, *PQCrypto 2020*, volume 12100 of LNCS, pages 35–50. Springer, 2020.
- [75] Gallager R. G., *Low-Density Parity-Check Codes*, 1963, MIT Press, Cambridge, MA.

[76]] P. A. Fouque et al., "Falcon: Fast-fourier lattice-based compact signatures over NTRU."

[77] <https://sphincs.org/>

[78] <https://sike.org/>

[79] Decoding One Out of Many - Nicolas Sendrier
. <https://eprint.iacr.org/2011/367.pdf>