Fifth interim report of deliverable V for CARS-87 project titled "Development of code-based PQC signature scheme and its security analysis" October 2024 to June 2025

Project Team at
Indian Institute of Information Technology Kalyani

01st October, 2025

Contents

List of Symbols

$\mathbf{C}\,[n, k]$	A linear code of length n and message length k
$\mathbf{C}\,[n, k, d]$	A linear code of length n, message length k and minimum distance d
$wt\,(e)$	Hamming-weight of vector e
$\mathbb{F}_q^n$	A vector space of n-tuples defined over finite field $\mathbb{F}_q$
$\mathbb{F}_q^{n,t}$	A subset of $\mathbb{F}_q^n$ where each tuple has Hamming weight t
$\{0,1\}^*$	A binary string of any length
$\{0,1\}^n$	A binary string of length n
$(a \mid\mid b)$	A string (or vector) formed by concatenating two strings (vectors) a and b
$\begin{bmatrix} A & B \end{bmatrix}$	A block matrix formed by concatenating two matrices A and B
I_k	The identity matrix of size $k \times k$
A^T	Transpose of matrix A
A^{-1}	Inverse of matrix A
$\lfloor x \rfloor$	The floor value of quantity x

Chapter 0

Project Overview

- **Name:** Development of code-based PQC signature scheme and its security analysis (PQCSS).
- **Funding Agency:** Centre for Artificial Intelligence and Robotics, Defense R&D Organization, DRDO, Govt. of India.
- **Sanction Letter No:** CAIR/SSD/CARS-87.
- **Sanctioned Amount:** Rs. $48,95,820$.
- **Start Date:** June 05, 2023.
- **Project Tenure:** 3 years.
- **Personnel:**
  - **PI:** Dr. Bhaskar Biswas.
  - **JRF:** Mr. Sayantan Chatterjee.

- **Deliverables:**

Del. No.	Deliverable Title	Deliverable Date
0	Project Start.	0 months
1	Procurement and report on theory of primitives and QC models (with up to date literature survey on PQC signature schemes).	3 months
2	Report of the design layout of the proposed signature scheme.	6 months
3	Theoretical analysis of the primitives.	12 months
4	Performance analysis of the proposed signature scheme for different parameters.	15 months
5	Report on theoretical cryptanalysis of the proposed system.	24 months
6	Software implementation (in ANSI C) of the proposed system.	27 months
7	Report on cryptanalysis (EUF-CMA security and MAXDEPTH computation or any other QC model analysis) using developed libraries.	30 months
8	Performance and security analysis report.	33 months
9	1. Project final report. 2. Software specification and design report.	36 months

0.1 Important Dates

- **Fund Received:** Rs. $15,55,932$ on June 05, 2023.
- **JRF Recruitment:**
  - **Date of JRF Advertisement:** June 20, 2023.
  - **Date of JRF Interview:** July 21, 2023.
  - **Date of Hiring of JRF:** August 01, 2023.
  - **No. of JRF(s) Recruited:** 01.
- **Procurement:**
  - **Purchase Initiated:** On June 12, 2023.
  - **Current Status:** Partially processed.

0.2 Review Meetings

0.2.1 Review Meeting 1: Online Meeting on Sept 01, 2023

Members Present:

Dr. Bhupendra Singh, Sc. F, CAIR, DRDO Mr. Susil Kumar Bishoi, Sc, E, CAIR, DRDO Dr. Debasish Bera, Asst. Prof, IIIT Kalyani Dr. Bhaskar Biswas, Asst Prof, IIIT Kalyani Mr. Sourabh Biswas, JRF, IIIT Kalyani, Mr. Sayantan Chatterjee, JRF, IIIT Kalyani.

MoM

1. Dr. Bhaskar Biswas gave a presentation on CARS-87 project "Development of code-based PQC signature scheme and its security analysis".

2. Mr. Susil Kumar Bishoi pointed out the changes to be made on the project report and were duly noted to be incorporated.

   - **Action Taken:** Proper changes were made to the report based on suggestions.

3. Dr. Bhupendra Singh suggested that quantum security models for RaCOSS and pqsigRM signature schemes be studied in detail.

   - **Action Taken:** We are concentrating more on the deliverables to be submitted (as committed on the project proposal) on a timely basis. If time permits we shall take over this topic in the future.

4. Issue was raised about the validity of the formula Gate-count , (G) x MAXDEPTH (D) = 2298 for AES-256. It was agreed that the issue needs to be checked in depth.

   - **Action Taken:** Still looking into the issue.

5. Mr. Susil Kumar Bishoi asked about raised the question how the quantum security gate count can viably be measured. Dr. Bhupendra Singh suggested to look into which quantum model is more effective for PQC. It is to be explored and discussed in future.

   - **Action Taken:** Still looking into the issue.

6. Dr. Bhaskar Biswas noted that algorithms are quantum safe because of the non-deterministic nature of the algorithms.

7. Issues regarding finances and project deliverables were discussed.

0.2.2 Review Meeting 2: Online Meeting on Feb 21, 2024

Members Present:

Dr. Bhupendra Singh, Sc. F, CAIR, DRDO Dr. Lexy Alaxander, Sc. E, CAIR, DRDO Dr. Debasish Bera, Asst. Prof, IIIT Kalyani Dr. Bhaskar Biswas, Asst Prof, IIIT Kalyani, Mr. Sayantan Chatterjee, JRF, IIIT Kalyani.

MoM

1. Mr. Sayantan Chatterjee described the proposed code based signature scheme.
2. Dr. Lexy Alexander pointed out a chosen-plaintext attack on the presented signature scheme.
    - **Action Taken:** The proposed signature scheme has been modified in order to avoid the attack.
3. Dr. Lexy Alexander pointed out that the size of matrix H_A in example 2.2.1 seems to be wrong.
    - **Action Taken:** The code-parameter was wrong, which has been corrected.
4. Dr. Bhupendra Singh and Dr. Lexy Alexander pointed out proper motivation is lacking from the new chapter.
    - **Action Taken:** New relevant sections have been added to the chapter.
5. Dr. Lexy Alexander commented that references of attacks (where available) should be added to each code-based signature schemes described in chapter 1.
    - **Action Taken:** References to major attacks (where available) are already provided for each scheme described.
6. Dr. Lexy Alexander commented that a summary of information (including references to attacks and parameter sizes) for each code-based scheme described in chapter 1 may be included in tabular form.
    - **Action Taken:** Table 1.1 has been updated with relevant information (where available).

0.2.3 Review Meeting 3: Online Meeting on May 14, 2024

Members Present:

Dr. Bhupendra Singh, Sc. F, CAIR, DRDO; Dr. Lexy Alaxander, Sc. E, CAIR, DRDO; Dr. Bhaskar Biswas, Asst. Prof., IIIT Kalyani; Mr. Sayantan Chatterjee, JRF, IIIT Kalyani.

MoM

1. Mr. Sayantan Chatterjee discussed topics like FDH signature scheme, KKS signature scheme and the modified novel signature scheme.

2. Dr. Lexy Alexander pointed out that in the key-generation phase in section 2.2 the dimensions of codes C_1 and C_2 need not be same as of code C.

    - **Action Taken:** The dimensions of codes C_1 and C_2 need not to be same as that of code C. Their dimensions is equal to the length of the hash of the message. The description of the protocol has been updated accordingly.

3. Dr. Bhupendra Singh and Dr. Lexy Alaxander asked for more clearer explanation for the parameters t_1 and t_2 in the description of the KKS scheme given in page 16, subsection 1.2.3.

    - **Action Taken:** The description has been updated.

4. Dr. Lexy Alexander pointed out the in the given scheme one-way hash function may be to used to avoid homomorphism attack.

    - **Action Taken:** The scheme has been modified accordingly.

5. Dr. Bhupendra Singh and Dr. Lexy Alaxander asked about the probability of attack if set J is repeated.

    - **Action Taken:** Theoretical expression for the likelihood of this attack has been given in section 3.2.

6. Dr. Lexy Alexander pointed out a typo in the definition of J in page 29, example 2.2.2.

    - **Action Taken:** The correct definition of J has been given.

7. Dr. Lexy Alexander mention the in the last line of page 29 the number of unknowns will be k instead of n.

    - **Action Taken:** The description of the attack has been modified.

8. Dr. Lexy Alexander pointed out that RODP of project is misstated in some places.

    - **Action Taken:** The report has been modified accordingly.

0.2.4 Review Meeting 4: First Yearly Review Meeting Held Online on June 13, 2024

Members Present:

Dr. Sugata Gangopadhyay, Prof., Dept. of Mathematics, IIT Roorkee; Dr. Bhupendra Singh, Sc. F, CAIR, DRDO; Dr. Lexy Alaxander, Sc. E, CAIR, DRDO; Dr. Alok Mishra, Sc. D, CAIR, DRDO; Dr. Bhaskar Biswas, Asst. Prof., IIIT Kalyani; Mr. Sayantan Chatterjee, JRF, IIIT Kalyani.

MoM

1. Dr. Bhaskar Biswas presented a 12-months review of work done till date.
2. Dr. Sugata Gangopadhyay pointed out that the notation of matrix G^* is wrong.
    - **Action Taken:** The definition of G^* has been more clearly stated.
3. Dr. Sugata Gangopadhyay pointed out that the statement "... as long as the signer knows the correct secret key G^*" is not clear.
    - **Action Taken:** The statement has been stated more clearly.
4. Dr. Sugata Gangopadhyay pointed out that weight of mG^* should be "greater than or equal to" t rather than "equal to" t
    - **Action Taken:** The condition has been changed accordingly. However, practically speaking, as the code-parameters are in our hand codes can be chosen in such a way that the minimum weight condition is satisfied by all codewords.
5. Dr. Lexy Alexander pointed out a typo in the definition code C_1 and C_2 in page 27 of the report.
    - **Action Taken:** The mistake has been corrected.

0.2.5 Review Meeting 5: Review Meeting Held Online on November 11, 2024

Members Present:

Dr. Bhupendra Singh, Sc. F, CAIR, DRDO; Dr. Lexy Alaxander, Sc. E, CAIR, DRDO; Dr. Bhaskar Biswas, Asst. Prof., IIIT Kalyani; Mr. Sayantan Chatterjee, JRF, IIIT Kalyani.

MoM

1. Sayantan Chatterjee presented on the topic of the 4th deliverable "Performance Analysis of the Proposed Signature Scheme for Different Parameters".

2. Dr. Lexy Alaxander pointed out a forgery attack on the proposed signature scheme.

    - **Comment:** The signature scheme has been modified to thwart this type of attack.

3. Dr. Bhupendra Singh remarked that the signature scheme must be implemented in a scalable way so that it can run efficiently even for large parameter values.

    - **Action Taken:** We are looking into ways to do this and it will be presented in the next deliverable.

4. Dr. Bhupendra Singh pointed out a mistake in page 31 of the report where the message m must be replaced by the hash of the message m. He also asked that the variables t_1 and t_2 should be replaced by d_1 and d_2 in page 29.

    - **Action Taken:** The corrections have been done.

5. Dr. Bhupendra Singh asked that security level data (for example 128-bit level security, etc.) should be there in the report.

    - **Action Taken:** Dr. Bhaskar Biswas pointed out that this will be included in the next deliverable.

Chapter 1

Code Based Signature Schemes

1.1 Outline

Public-key cryptography is a type of cryptosystem that utilizes two separate keys for encryption and decryption. The key used for encryption is known as the public key (hereafter represented as $\mathcal{K}_{Pub}$). As the name suggests, the content of the public key is open to everybody, allowing anyone to encrypt messages for a user as long as they possess the user's public key. Conversely, the key used for decryption is known as the private or secret key ($\mathcal{K}_{Priv}$) and is only known to the key owner. Encryption in a public-key cryptosystem takes a message and the public key of the receiver as input and outputs a *cipher-text*. Decryption takes a cipher-text and the receiver's private-key as input and outputs a *plain-text*. A correctly implemented public-key cryptosystem has the property:

$$\mathbf{Dec}\left(\mathbf{Enc}\left(m, \mathcal{K}_{Pub}\right), \mathcal{K}_{Priv}\right) = m \tag{1.1}$$

Here $\mathbf{Enc}\left(\right)$ and $\mathbf{Dec}\left(\right)$ represent the encryption and decryption functions, respectively, and m represents the message.

However, providing privacy is not the sole utility of a public-key cryptosystem. Public-key cryptosystems can also offer other security services, such as *authentication* and *non-repudiation* through the use of *digital signatures*.

A digital signature is a securely generated unique token mapped to a user that, like a real signature, authenticates any document (or message) generated by the user. Such a signature also provides non-repudiation, meaning that once a message is signed and sent by a user, the user cannot deny having sent the message at a later time.

Digital signatures can be created in various ways, with one of the most popular methods being the use of public-key cryptography. In this scheme, a one-way secure hash of the message is first taken. Then, the hash value is decrypted by the message-generator using their private key, and the decrypted hash value (known as the signature) is attached to the end of the message. Both the message and its signature are then sent. The receiver of the message encrypts the received signature using the sender's public key and verifies the authenticity of the message by comparing the encrypted value with the hash of the

received message, ensuring that the message was indeed sent by the purported sender and was not modified during transmission. The general scheme is described below.

- **Setup:** The sender of the message generates a pair of keys $\mathcal{K}_{Pub}$ and $\mathcal{K}_{Priv}$. The public key $\mathcal{K}_{Pub}$ is kept openly available. The private key $\mathcal{K}_{Priv}$ is kept securely undisclosed by the sender. The sender also chooses a cryptographic hash function $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^l$, where l is the size of the first input (i.e. cipher-text length) of the **Dec** () function.

- **Message Signing:** Let m be the message which needs to be signed. The sender creates the signature of the message as

$$s = \mathbf{Dec}\left(\mathcal{H}\left(m\right), \mathcal{K}_{Priv}\right) \tag{1.2}$$

  The sender then sends the message-signature pair (m, s) to the intended receiver.

- **Message Verification:** The receiver receives the message-signature pair $\left(m^{'}, s^{'}\right)$. The receiver verifies the authenticity of the message by checking whether

$$\mathcal{H}\left(m^{'}\right) \stackrel{?}{=} \mathbf{Enc}\left(s^{'}, \mathcal{K}_{Pub}\right) \tag{1.3}$$

The scheme given above is also known as Full-Domain Hash (FDH). While code-based cryptosystems are public-key cryptosystems, FDH cannot be straightforwardly applied to code-based cryptography. As will be seen in the coming sections, this is because not all messages (or hash values) can be decrypted in code-based cryptography.

This chapter provides an overview of all the prominent code-based signature schemes that have been developed over the years. Section 1.2 summarizes the major code-based signature schemes, while Section 1.3 discusses the different hard problems on which the security of these signatures depend. Subsection 1.3.1 lists the values of the quantum security parameter MAXDEPTH as proposed by NIST. Lastly, Section 1.4 explores the advantages and disadvantages of using code-based cryptography.

1.2 Various Proposals

1.2.1 Xinmei and Modified Xinmei Signature Schemes

The first code-based cryptosystem was proposed by Robert J. McEliece in 1978 [21]. In 1986, Harald Niederreiter published another cryptosystem based on McEliece's work [23]. Subsequently, it was demonstrated that both the McEliece cryptosystem and the Niederreiter cryptosystem are equivalent [20].

However, the first signature scheme based on error-correcting codes emerged only in 1990. This scheme, developed by Wang Xinmei, employed linear codes for both signature generation and error correction [35]. The Xinmei scheme is described in detail below.

- **Setup:** The signer initiates the process by selecting a $[n, k, 2t_A - 1]$ Goppa code denoted as C_A. A $k \times n$ generator matrix G_A and a $(n-k) \times n$ parity check matrix H_A for code C_A are chosen. Additionally, the signer generates two invertible matrices: a $n \times n$ matrix P_A and a $k \times k$ matrix S_A. Another matrix G_A^* of size $n \times k$ is computed, where $G_A G_A^* = I_k$, the $k \times k$ identity matrix. Several quantities are then calculated as follows:

$$\begin{aligned} J_A &= P_A^{-1} G_A^* S_A^{-1} \\ W_A &= G_A^* S_A^{-1} \\ T_A &= P_A^{-1} H_A^T \end{aligned} \tag{1.4}$$

  Here, the dimensions of J_A, W_A, and T_A are $n \times k$, $n \times k$, and $n \times (n-k)$ respectively.

- **Public Keys:** The public keys are comprised of J_A, W_A, T_A, H_A, t_A, and $t^{'}$, where $t^{'} < t_A$.

- **Private Keys:** The signer retains S_A, G_A, G_A^* and P_A as private keys.

- **Message Signing:** To sign a k-bit message m_j, a random binary error vector e_j of size n and weight $t^{'}$ is chosen. The signature is then computed as:

$$c_j = (e_j + m_j S_A G_A) P_A. \tag{1.5}$$

  The resulting message-signature pair (m_j, c_j) is then sent to the receiver.

- **Message Verification:** Upon receiving the potentially error-affected signature $c_j^{'}$, the receiver conducts signature verification in four steps:
  - **Step 1:** The receiver computes the syndrome of $c_j^{'}$ as $S\left(c_j^{'}\right) = c_j^{'} T_A = e_j^{'} H_A^T$.
  - **Step 2:** The receiver decodes $S\left(c_j^{'}\right)$ using Berlekamp-Massey algorithm to obtain $e_j^{'}$. If $wt(e_j^{'}) = t^{'}$, the receiver concludes that $c_j^{'} = c_j$. Otherwise the receiver requests re-transmission.
  - **Step 3:** The receiver then calculates $c_j J_A = e_j G_A^* S_A^{-1} + m_j$.
  - **Step 4:** In the final step, the receiver computes $c_j J_A + e_j W_A$ and verifies whether it equals m_j or not.

In 1992, Harn and Wang discovered that Xinmei's scheme is vulnerable to a selective forgery attack due to the linearity of the signature operations [13]. Suppose a counterfeiter has access to two message-signature pairs (m_1, c_1) and (m_2, c_2). Then,

$$\begin{aligned} Sig(m_1) + Sig(m_2) &= (e_1 + m_1 S_A G_A) P_A + (e_2 + m_2 S_A G_A) P_A \\ &= ((e_1 + e_2) + (m_1 + m_2) S_A G_A) P_A \\ &= Sig(m_1 + m_2). \end{aligned} \tag{1.6}$$

Hence, if the weight of $e_1 + e_2$ is less than or equal to t_A, the signature of message $m_1 + m_2$ can be forged using the signatures of m_1 and m_2.

To thwart the aforementioned attack, Harn and Wang proposed the use of a hash function before signing the messages. Their modified scheme, henceforth referred to as the modified Xinmei scheme, is outlined as follows.

- **Setup:** Similar to the Xinmei signature scheme, except that P_A is now a $k \times k$ permutation matrix. Additionally, the signer publishes a cryptographically strong one-way hash function denoted as $\mathcal{H}()$, which takes an l-bit message and produces an n-bit hash, where $l > n$.
- **Public Keys:** Remains the same as the Xinmei scheme.
- **Private Keys:** Remains the same as the Xinmei scheme.
- **Message Signing:** The n-bit signature c_j of an l-bit message m_j is computed as:

$$c_j = \mathcal{H}(m_j) S_A G_A P_A \tag{1.7}$$

  The resulting message-signature pair (m_j, c_j) is then transmitted. When the signature reaches the receiver, it may be subject to an error e_j, causing the receiver to receive $c'_j = e_j + c_j$.
- **Signature Verification:** The verification process is conducted similarly to the Xinmei scheme, with the exception that instead of matching m_j, the hash value $\mathcal{H}(m_j)$ is compared.

By incorporating these measures, the adapted Xinmei scheme aims to bolster its defense against attacks and ensure the security of the signature process. However, in the years that followed, several security vulnerabilities were uncovered in both the original Xinmei scheme and its modified version [1, 19, 32]. As a result, these schemes have lost their practical viability as implementable signature solutions.

1.2.2 Alabbadi and Wicker's Signature Scheme

In response to the security gaps in Xinmeni and modified Xinmei schemes, Mohssen Alabbadi and Stephen B. Wicker introduced their own signature scheme in 1993 [2]. Their signature scheme functions as follows:

- **Setup:** The initial step involves selecting a $[n, k, d]$ Goppa code C_A with an error-correcting capacity of $t_A = \lfloor (d-1)/2 \rfloor$. Subsequently, a $k \times n$ generator matrix G_A and a $(n-k) \times n$ parity-check matrix H_A are chosen for code C_A. Furthermore, an $n \times k$ matrix G_A^* is derived, satisfying the equation

$$G_A G_A^* = I_k \tag{1.8}$$

An invertible $n \times n$ binary matrix P_A is generated, leading to the calculation of the $n \times k$ matrix

$$G_A^{'} = P_A^{-1} G_A^* \tag{1.9}$$

and the $n \times (n-k)$ matrix

$$H_A^{'} = P_A^{-1} H_A^T \tag{1.10}$$

Additionally, a $n \times l$ binary matrix R_A of rank n is chosen, where $n < l$. The corresponding $l \times n$ matrix R_A^* is determined which satisfies

$$R_A R_A^* = I_n \tag{1.11}$$

Finally, the signer selects a nonlinear one-way function denoted as $f(x, y)$, where x is a k-bit binary string and y is an n-bit binary string. The output of the function f is a k-bit value. This function f is made public.

- **Public Keys:** The public keys are $G_A^{'}$, $H_A^{'}$, H_A, R_A^*, t_A, $t_A^{'}$, where $t_A^{'} < t_A$.
- **Private Keys:** G_A, P_A, G_A^* and R_A are the private keys.
- **Message Signing:** For a k-bit message m_j, a random binary error z_j of length n and weight $t_A^{'}$ is selected. Furthermore, a random l-bit long vector, e_j, of arbitrary weight is also chosen. The l-bit signature c_j is calculated as

$$c_j = \left[\left(z_j + \left[f\left(m_j, z_j\right) + z_j G_A^*\right] G_A\right) P_A + e_j R_A^*\right] R_A + e_j \tag{1.12}$$

- **Message Verification:** The receiver initially computes $v_j = c_j R_A^*$, leading to:

$$\begin{aligned} v_j &= c_j R_A^* \\ &= \left(z_j + \left[f\left(m_j, z_j\right) + z_j G_A^*\right] G_A\right) P_A \end{aligned} \tag{1.13}$$

Subsequently, the receiver calculates $s_j = v_j H_A^{'}$, resulting in:

$$\begin{aligned} s_j &= v_j H_A^{'} \\ &= z_j H_A^T \end{aligned} \tag{1.14}$$

This value is then decoded using the Berlekamp-Massey algorithm to determine z_j. Following this, the receiver computes $s_j^{'} = v_j G_A^{'}$:

$$\begin{aligned} s_j^{'} &= v_j G_A^{'} \\ &= z_j G_A^* + f\left(m_j, z_j\right) + z_j G_A^* \\ &= f\left(m_j, z_j\right) \end{aligned} \tag{1.15}$$

The message is verified if $s_j^{'}$ matches $f\left(m_j, z_j\right)$.

This scheme designed by Alabbadi and Wicker presents an alternative approach to signature security while addressing the vulnerabilities found in previous schemes. However in 1993 Alabadi and Wicker discovered multiple security loopholes in their own scheme which made it vulnerable to universal forgery attack [3]. Moreover, they also showed that all the previous signature schemes are vulnerable to the same universal forgery.

1.2.3 Kabatianskii-Krouk-Smeets (KKS) Signature Scheme

In 1997, G. Kabatianskii, E. Krouk, and B. Smeets introduced a novel signature based on the Niederreiter cryptosystem-like signature scheme (referred to as the KKS scheme) [17]. This marked the first utilization of code-based cryptography in a Full Domain Hash (FDH) scheme. The Niederreiter signature scheme operates as follows:

- **Setup:** Let $C_A[n, k, d]$ be an efficiently decodable linear code having parity check matrix H_A of size $(n-k) \times n$. A random invertible binary matrix A with dimensions $(n-k) \times (n-k)$ is generated. Also, a permutation matrix P of size $n \times n$ is created. The matrix $H_{Pub} = AH_AP$ with dimensions $(n-k) \times n$ is then computed.

- **Public Keys:** The public keys consist of H_{Pub} and the parameter $t = \lfloor \frac{d-1}{2} \rfloor$, representing the error-correcting capacity of the code C_A.

- **Private Keys:** The matrices A and P along with the parity check matrix H_A are private keys.

- **Message Signing:** The message m belongs to set of all correctable syndromes of H_{Pub}, i.e. $m \in \mathcal{M}$ where $\mathcal{M} = \{s : H_{Pub}e^T = s \text{ for some } e \in \mathbb{F}_2^{n,t}\}$. Here $\mathbb{F}_q^{n,t}$ represents all q-arry strings of length n and weight t. Then the signature of the message m is the error e. Let $D_A()$ represent the efficient syndrome decoding algorithm of C_A. Then the signer calculates the signature of message m as

$$e = D_A\left(A^{-1}m\right)P \tag{1.16}$$

- **Signature Verification:** The receiver of the message-signature pair (m, e) verifies the signature by first checking that $wt(e) = t$ and then the equation

$$m = H_{Pub}e^T \tag{1.17}$$

  is satisfied.

The security of this signature scheme derives from the complexity of the syndrome decoding problem, which is known to be NP-complete [31, 4]. However, the scheme encounters a significant issue: not all messages are guaranteed to be within the set $\mathcal{M}$ of correctable syndromes. To address this, a solution involves creating a mapping from the message set to the set of correctable syndromes, but this approach is costly due to the storage and transfer requirements. Each entry in the mapping requires $\log_2 \binom{n}{t}$ bits, resulting in a total size of $2^k \log_2 \binom{n}{t}$ bits for 2^k messages.

Kabatianskii, Krouk, and Smeets proposed a Neiderreiter-based signature scheme that avoids the need for decoding during the signing process. In this version, the code C_A can be any random linear code derived from the parity check matrix H_A, provided its minimum distance d is sufficient to make syndrome decoding challenging. The signer establishes a direct mapping f from the message space to the signature space: $f : \mathbb{F}_2^k \to \mathbb{F}_2^{n,t}$.

The message $m \in \mathbb{F}_2^k$ is mapped to a corresponding signature $s \in \mathbb{F}_2^{n,t}$ via $s = f(m)$. Additionally, another mapping ξ is created from the message set to the set of correctable syndromes: $\xi(m) = H_A f(m)^T$.

In this scheme, f remains private while ξ is made public. The signer sends a message-signature pair (m, s) where $s = f(m)$. The recipient verifies the signature by confirming that the weight of s falls within some range $t_1 \leq wt(s) \leq t_2$ and by validating the equation:

$$\xi(m) = H_A s^T \tag{1.18}$$

It is important to note that the proposed scheme cannot operate with a random map f. While this enhances the security of the signature scheme, it requires a substantial amount of space to store f and ξ, as previously discussed. Kabatianskii, Krouk, and Smeets introduced three versions of the signature scheme, each containing three distinct mappings f. The general KKS signature scheme is outlined as follows:

- **Setup:** Similar to before let H_A be a random parity check matrix of size $(n-k) \times n$ corresponding to linear code $C_A[n, k, d]$. Let V be another $[n', k]$ linear code with $n' < n$ and generator matrix $G' = [g_{ij}]_{k \times n'}$. Let the code V be such that any codeword of V has Hamming weight in the range t_1 and t_2, where $0 < t_1 \leq t_2$ and $d > 2t_2$. Let J be a subset of $\{1, 2, \ldots, n\}$ of cardinality n'.

  The signer computes $G^* = \left[g_{ij}^*\right]_{k \times n}$, where $g_{ij}^* = g_{ij}$ for $j \in J$ and $g_{ij}^* = 0$ otherwise. The signer additionally constructs $H_A(J)$ by selecting n' columns from H_A, each with an index in J. The signer now calculates

  $$F = H_A(J) G'^T \tag{1.19}$$

- **Private Keys:** The private keys include the subset $J \subset \{1, 2, \ldots, n\}$ and the $k \times n'$ matrix G'.

- **Public Keys:** The public keys comprise the $(n-k) \times k$ matrix F and the $(n-k) \times n$ parity check matrix H_A.

- **Message Signing:** The signature of a message m is computed as $f(m) = mG^*$. Notably, the length of message m is k, while its signature length is n. Moreover, the signature weight falls between t_1 and t_2.

- **Signature Verification:** The recipient of the message-signature pair (m, s) verifies the signature by confirming that $t_1 \leq wt(s) \leq t_2$ and by validating the equation:

  $$Fm^T = H_A s^T \tag{1.20}$$

  This equation holds because $H_A G^{*T} = H_A(J) G'^T$.

The KKS signature scheme was the first FDH signature scheme which seemed to avoid all the security loopholes of the signature schemes that came before it. However, a very simple key recovery attack on KKS scheme was discovered by Pierre-Louis Cayrel, Ayoub Otmani and Damien Vergnaud in 2007 [8].

1.2.4 Barreto-Misoczki-Simplico (BMS) Signature Scheme

In 2011, Barreto, Misoczki and Simplico presented a modification of KKS signature scheme by adding random error to the signature [7]. The BMS scheme works as follows [25]:

- **Setup:** The setup of BMS scheme is similar to that of KKS scheme. The signer S first selects parameters N, K, n, k, t_1 and t_2 such that $t_1 < t_2$. The values of these parameters are chosen according to desirable security level. S then chooses a $(N-K) \times N$ parity-check matrix H for a $C[N, K]$ code. S also chooses a secret $C_h[n, k]$ code having $k \times n$ generator matrix G such that all codewords $c \in C_h$ have weight between t_1 and t_2. The set $J \subset \{1, 2, \ldots, N\}$ of cardinality n is then chosen by S. Let H_J be the $(N-K) \times n$ matrix which is formed by taking the i_{th} column of H for all $i \in J$. S computes $F = H_J G^T$.

  S also determines a one-way hash function $\mathcal{H} : \{0,1\}^* \times \mathbb{F}_2^{N-K} \to \mathbb{F}_2^k$ and makes it public.

- **Private Keys:** The private keys are the generator matrix G and the set J.

- **Public Keys:** The public keys comprise of the matrices H and F.

- **Message Signing:** : To sign a message $m \in \{0,1\}^*$ the signer S follows the following steps:

  1. Choose a random vector e of length N and weight n.
  2. Calculate $h = \mathcal{H}\left(m, He^T\right)$.
  3. Compute $y = hG$.
  4. If $\text{supp}(y) \subset J$ then signature of m is $(h, y+e)$.
  5. Else go back to step 2.

  Once signature is calculated S sends the message-signature triple $(m, h, s = y + e)$ to the receiver.

- **Signature Verification:** Upon receiving the message signature triple (m, h, s), the receiver first checks that $wt(s) \leq 2n$ (the weight of e is n, the weight of y is maximum n, thus weight of $s = y + e$ is smaller then or equal to $2n$) and then checks whether

$$h \stackrel{?}{=} \mathcal{H}\left(m, Hs^T + Fh^T\right) \tag{1.21}$$

The BMS scheme is also vulnerable to the key-recovery attack given in [8] and hence can be used only as a one-time signature scheme (i.e. every message must be signed using a new key) [7]. In 2011, Ayoub Otmani and Jean-Pierre Tillich presented an additional attack on both KKS and BMS signatures that doesn't necessitate any known message-signature pairs for successful execution [25].

1.2.5 Random Code-based Signature Scheme (RaCOSS)

In 2017, NIST initiated a call for proposals for new post-quantum cryptosystems. One of the KKS type signature schemes put forth during the first round of this initiative was the Random Code-based Signature Scheme (RaCOSS), proposed by Roy et al. [29]. The scheme's workings are elaborated as follows:

- **Setup:** The signer, denoted as S, selects parameters: n, k, ω, and γ. The original RaCOSS specification recommended $n = 2400$, $k = 2060$, $\omega = 48$, and $\gamma = 0.07$. Additionally, S computes:

$$\begin{aligned}\rho &= \left\lfloor \frac{1}{2}\left(1 - \left(1 - \frac{2\omega}{n}\right)^{\gamma\omega}\right) \cdot 1000 \right\rfloor / 1000 \\ \theta &= 12n\rho\,(1-\rho) \\ w &= \lfloor \gamma\omega \rfloor\end{aligned}$$

  A random $(n-k) \times n$ binary matrix H is generated, with each element having an equal probability of being 1 or 0. Similarly, an $n \times n$ binary matrix X is created, where the probability of any element being 1 is ω/n. The matrix $F = HX$, with dimensions $(n-k) \times n$, is computed.

  Additionally, the signer specifies a one-way hash function denoted as $\mathcal{H}(m, p, t)$, which takes a byte-string m and outputs a hash of length p and weight t. The method of constructing $\mathcal{H}$ is detailed in [29].

- **Private Key:** The private key is represented by the matrix X.

- **Public Key:** The public keys are matrices F and H.

- **Message Signing:** Let $enc(H)$ denote bit string constructed by concatenating the rows of matrix H. Let $byte(v)$ be the byte string corresponding to the bit-string v. If necessary, 0s are appended to v to make its size a multiple of 8. The following steps are performed by S to sign a message m.

  1. Construct a random vector y of length n, with each bit having a probability ρ of being 1.
  2. Compute $v = Hy^T$.
  3. Formulate $b = (byte(v) \,||\, byte(m) \,||\, byte(enc(H)))$.
  4. Calculate $c = \mathcal{H}(b, n, w)$, producing a bit-string of length n and weight w.
  5. Compute $z = Xc + y$.
  6. The signature of message m is (z, c).

  The signer sends the message-signature triple (m, z, c) to the recipient.

- **Signature Verification:** Upon receiving the message-signature triple (m, z, c) the verifier undertakes the following steps to verify the signature:

1. If the Hamming weight of z is greater than or equal to θ then the signature is deemed invalid. Otherwise, proceed to step 2.
2. Calculate $v = Hz + Fc$.
3. Compute $b = (byte\,(v)\ ||\ byte\,(m)\ ||\ byte\,(enc\,(H)))$ and $c^{'} = \mathcal{H}\,(b, n, w)$.
4. If $c^{'}$ equals c the signature is valid; otherwise, it's not.

The RaCOSS signature scheme is afflicted by implementation errors, weaknesses in the hash function, mathematical inaccuracies, and other security loopholes [34], which were thoroughly expounded upon in [16]. In response, Roy et al. revised the signature scheme, resulting in RaCOSS-r [28], which was published in the year 2018. However, the RaCOSS-r scheme is not exempt from security flaws, as outlined in [34]. As a consequence, the RaCOSS signature scheme did not meet the criteria for advancement to the next NIST PQC round.

1.2.6 Curtois-Finiasz-Sendrier (CFS) Signature Scheme

An alternative to KKS type signature schemes was introduced in 2001 by Nicolas Curtois, Matthieu Finiasz and Nicolas Sendrier [9]. Their scheme (henceforth denoted as CFS signature scheme) is a FDH scheme based on Neiderreiter signature scheme . The CFS scheme is probabilistic in nature. A slight modification was done to the scheme by Leonard Dallot [10]. The modified scheme works as follows:

- **Setup:** Like Neiderreiter signature scheme an efficiently decodable linear code $C_A\,[n, k, d]$ is chosen with parity check matrix H_A of size $(n-k)\times n$. A random invertible binary matrix A of dimension $(n-k)\times(n-k)$ is generated. Also a permutation matrix P of size $n \times n$ is created. The matrix $H_{Pub} = AH_AP$ with dimension $(n-k)\times n$ is then computed. In addition a one-way hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{n-k}$ is chosen and made public.
- **Public Keys:** The public keys consist of H_{Pub} and the parameter $t = \lfloor \frac{d-1}{2} \rfloor$, representing the error-correcting capacity of the code C_A.
- **Private Keys:** The matrices A and P along with the parity check matrix H_A are private keys.
- **Message Signing:** Let m be the message whose signature has to be generated. The following steps are followed [10]:
  1. Choose a random token $i \in \{1, 2, \ldots, 2^{n-k}\}$.
  2. Try to decode $A^{-1}\mathcal{H}\,(m\ ||\ i)$, i.e. find a $x \in \mathbb{F}_2^{n,t}$ such that $A^{-1}\mathcal{H}\,(m\ ||\ i) = H_A x^T$.
  3. If no such x is found then go to Step 1.

4. Else $s = xP$ is the signature of m. Send message-signature pair (m, s, i) to the receiver.

- **Signature Verification:** Signature verification is carried out by checking whether $wt(s) = t$ and

$$H_{Pub} s^T \stackrel{?}{=} \mathcal{H}(m \,||\, i) \tag{1.22}$$

In 2007 Dallot provided a concrete security proof of the CFS scheme under random oracle model [10]. However, the CFS scheme has not seen widespread adoption due to the probabilistic nature of the algorithm which makes it extremely slow as multiple decodings has to be carried out for different values of the token i [33].

1.2.7 Low Density Generator Matrix (LDGM) code-based Signature Scheme (BBCRS)

In 2013, Baldi et al. presented a variation of the CFS scheme that relies on Low-Density Generator Matrix (LDGM) codes [6]. An LDGM code is a linear code with a sparse generator matrix. The matrices of this code, henceforth referred to as the BBCRS code, are also quasi-cyclic, enhancing the efficiency of key storage. The BBCRS scheme is outlined below:

- **Setup:** An LDGM code $C[n, k]$ is chosen. Let H be the $(n-k) \times n$ parity check matrix of C. H is chosen to be in standard form, i.e. $H = \begin{bmatrix} X^T & I_r \end{bmatrix}$, where $r = (n-k)$ and I_r is the $r \times r$ identity matrix. Two invertible matrices Q and S of sizes $(n-k) \times (n-k)$ and $n \times n$ respectively are chosen. The matrix $H^{'} = Q^{-1} H S^{-1}$ is calculated.

  The matrix S is chosen to be sparse-matrix with average row and column weight of $w_S \ll n$. The matrix Q is chosen such that $Q = R + T$ where matrix R is a $(n-k) \times (n-k)$ low-rank (of rank $z < (n-k)$) dense-matrix and T is a sparse-matrix with column-weight of m_T. In order to construct R such that R has rank z two $z \times (n-k)$ matrices a and b are chosen such that $R = a^T b$.

  Furthermore, a hash function denoted as $\mathcal{H}$ is chosen, alongside another function $\mathcal{F}$ that takes a hash as input and produces a bit-string of length $n-k$, with a weight of $w \ll n-k$. The construction procedure for $\mathcal{F}$ is outlined in [6].

  To save storage space the matrices H, Q, S. etc. are designed to be quasi-cyclic.

- **Private Keys:** The private keys constitute of the matrices H, Q, S, R, T, a and b.

- **Public Keys:** The matrix $H^{'}$ is the public key.

- **Message Signing:** In order to sign a message m the sender carries out the following steps:

1. Calculate $s = \mathcal{F}(\mathcal{H}(m))$, of length $n-k$ and weight w. Verify whether $b \cdot s = 0_{z\times 1}$. If not then go to step 2. Else go to step 5.
2. Initialize a counter l.
3. Calculate $s = \mathcal{F}(\mathcal{H}(m \mid\mid l))$
4. If $b \cdot s \neq 0_{z\times 1}$ then increment l and go to step 3.
5. Calculate $s^{'} = Q \cdot s$ and $e = \begin{bmatrix} 0_{1\times k} & s^{'T} \end{bmatrix}$.
6. Finally select a codeword $c \in C$ of low weight w_c and compute the signature as $e^{'} = (e + c) \cdot S^T$.

After completing the above steps the sender sends the message-signature pair $\left(m, e^{'}\right)$ to the receiver.

- **Signature Verification:** After receiving the message signature pair $\left(m, e^{'}\right)$ the receiver carries out the following steps:
  1. Check whether $wt\left(e^{'}\right) \leq (m_T w + w_c)\, m_S$. If not then reject the signature.
  2. Check whether $wt(\mathcal{F}(\mathcal{H}(m))) = w$. If not then reject the signature.
  3. If $H^{'} \cdot e^{'T} = \mathcal{F}(\mathcal{H}(m))$ then accept the signature; otherwise, reject.

The BBCRS scheme was designed by its authors to be an efficient and secure signature scheme. However, in 2016 Aurélie Phesso and Jean-Pierre Tillich found that due to an existing correlation between the bits of the signature an 80-bit security of the BBCRS scheme can be broken with access to just $100,000$ signatures signed with the same key [27].

1.2.8 Escher code-based Signature Scheme

In 2014, Gligoroski et al. presented a novel FDH signature scheme that utilized a new type of code referred to as "McEliece in the world of Escher" (referred to as Escher code in this report) [12]. The Escher code does not characterize the error vector by Hamming weight; instead, it employs the concept of the *error set* as a newly defined metric. Within this scheme, decoding is non-unique, and the notion of *list decoding* comes into play. By combining error sets with list decoding, the scheme ensures that all words can be decoded automatically with overwhelming probability. As a result, a message intended for signing can be promptly decrypted (using the algorithm defined in [12]) and then attached as a signature. This contributes to the efficiency of signature generation and verification.

However, in 2016, Dustin Moody and Ray Perlner discovered that Gilgoroski et. al.'s scheme for 80-bit security could be easily compromised using a modified version of Stern's Information Set Decoding (ISD) algorithm [22]. To counter Moody and Perlner's attack, the size of the parameters of the Escher signature scheme must be increased to an extent where the entire scheme becomes impractical.

1.2.9 The pqsigRM Signature Scheme

In 2017, Lee et al. introduced a signature scheme similar to CFS, but this time employing punctured Reed-Muller (RM) codes [18]. This scheme, referred to as pqsigRM, aimed to rectify certain weaknesses inherent in the CFS scheme, such as the need for excessively large keys to ensure sufficient security and the decoding time's dependence on the factorial of error weight. The pqsigRM was submitted to the first round of NIST PQC call-for-proposal. The scheme utilizes the *closest coset* decoding algorithm to efficiently decode nearly all words of syndrome length. By incorporating punctured RM codes, the scheme aimed to mitigate existing attacks on RM-based cryptosystems.

However, subsequent comments highlighted concerns regarding the security of the scheme [26]. Doubts were raised concerning the security of the puncturing algorithm employed in pqsigRM. Following multiple iterations and modifications of the original algorithm, the scheme ultimately failed to meet security criteria and was consequently deemed not viable.

As can be seen from this section, various code based signature schemes have been proposed throughout the year. However, no clear choice exist as far as efficient, secure and practical signature scheme is concerned.

1.3 Security

The security of code-based cryptography, in general, and code-based signature schemes, in particular, is dependent on the hardness of certain problems in the domain of error-correcting codes. Hardness of a problem refers to the absence of sub-exponential time algorithms for solving it. The code-based signature schemes discussed in the previous section are designed such that breaking them implies that one or more underlying hard problems can be efficiently solved. Therefore, the fact that no efficient algorithms exist for the hard problems implies that the signature schemes are secure.

Below, some of the hard problems used in code-based signature schemes are defined in terms of their input and output.

- Matrix Factorization (MF) Problem:
  - *Input:* A matrix M where $M = SP$ for two product compatible matrices S and P.
  - *Output:* The matrices S and P.
- Bounded Decoding (BD) Problem:
  - *Input:* A random code $C[n, k, d]$ characterized by its $k \times n$ generator matrix G along with the vector $r = mG+e$ for some $m \in \mathbb{F}_2^k$ and $e \in \mathbb{F}_2^{n,t}$ where $t \leq \lfloor \frac{d-1}{2} \rfloor$.
  - *Output:* The vector m.
- Syndrome Decoding (SD) Problem:

Signature Scheme	Parameter Size	Type of Scheme	Attacks
Xinmei signature scheme [35]	$(n-k) > 100$	NA	[13, 1, 19, 32]
Modified Xinmei signature scheme [13]	NA	NA	[1, 19, 32]
Alabbadi and Wicker's signature scheme [2]	NA	NA	[3]
Kabatianskii-Krouk-Smeets signature scheme (KKS) [17]	$n = 1100$ $k = 335$ $t_1 = t_2 = 8$ Size of public key is 3×10^5 bits Size of private key is 2004 bits	FDH	[8, 25]
Barreto-Misoczki-Simplico signature scheme (BMS) [7]	For 256 bit security: $n = 37274$ $k = 512$ $\mid J \mid = 6761$ Size of signature is 11961 bits	Single-key KKS type	[8, 25]
RaCOSS [29]	For 177 bits security: $n = 2400$ $k = 2060$ $\omega = 48$ $\gamma = 0.07$	KKS type	[16, 34]
Curtois-Finiasz-Sendrier signature scheme (CFS) [9]	$n = 65536$ $k = 65392$ $t = 9$	FDH	NA
LDGM code-based signature scheme (BBCRS) [6]	For 160 bits security: $n = 46000$, $k = 16000$ $w_S = 20$, $z = 2$ $m_T = 1$ $w = 29$ Signature size is 1683 KiB	CFS	[27]
Escher code-based signature scheme [12]	For 140 bit security: $k = 786$ Size of public key is 75 KiB	CFS	[22]
pqsigRM signature scheme [18]	For 256 bit security: $n = 8192$, $k = 4096$ $d_{min} = 128$ $w = 1367$	CFS	[26]

Table 1.1: Summary of all code-based signature schemes

- *Input:* A code $C[n, k, 2t-1]$ characterized by its $(n-k) \times n$ parity check matrix H and syndrome $s \in \mathbb{F}_2^{n-k}$ where $s^T = He^T$ for some $e \in \mathbb{F}_2^{n,t}$.
- *Output:* The vector e.

- Goppa Parameterized Bounded Decoding (GPBD) Problem [10]:
  - *Input:* A $(n-k) \times n$ matrix H and a syndrome $s \in \mathbb{F}_2^{n-k}$.
  - *Output:* A vector $e \in \mathbb{F}_2^n$ such that $wt(e) = \frac{n-k}{\log_2 n}$ and $He^T = s^T$.

- Goppa Code Distinguishing (GD) Problem [10]:
  - *Input:* A $(n-k) \times n$ matrix H which may either be a parity check matrix of a Goppa code or parity check matrix of a random liner code with equal probability.
  - *Output:* Correctly output whether H is a parity check matrix of a Goppa code or the parity check matrix of a linear code.

The decision versions of BD and SD problems were proved to be NP-complete in 1978 [31, 4]. The BD problem states that given an error-affected codeword and the generator matrix of a random linear code, it is NP-complete to decode the message (or equivalently, the error). The SD problem is the equivalent problem of finding the error given the syndrome and the parity check matrix of a random linear code.

The GPBD problem is a special case of SD problem. A Goppa code is a linear code whose parameters depend on integers m and t. Given m and t, a Goppa code has length $n = 2^m$ and dimension $k = n - mt$. The GPBD problem states that decoding is hard for random linear code with Goppa code like parameters. The problem was introduced in [11, 30].

The GD problem states that it is NP-complete to distinguish a Goppa code from a random linear code [30].

The Xinmei [35], modified Xinmei [13], and Alabbadi and Wicker signature schemes [2] depend on the hardness of the MF and SD problems for their security. The SD problem is also the underlying hard problem for the KKS signature scheme [17] and similar signature schemes such as BMS [7] and RaCOSS [29]. The security of the CFS signature scheme [9] was shown to be dependent on the hardness of the GPBD and GD problems in [10]. The BBCRS scheme [6], LDGM scheme [6], Escher scheme [12], and pqsigRM schemes [18] all utilize the BD and SD problems for their security.

1.3.1 Quantum Security Strength Categories as given by NIST

On July 17, 2023, NIST made an announcement regarding a call for "Additional Digital Signature Candidates for PQC Standardization." As part of this call for proposals, NIST introduced several security strength categories that will be used to assess the effectiveness of the proposed signature schemes. Among the parameters introduced by NIST, there is one known as MAXDEPTH.

AES-128	2^{157}/MAXDEPTH quantum gates or 2^{143} classical gates
SHA3-256	2^{146} classical gates
AES-192	2^{221}/MAXDEPTH quantum gates or 2^{207} classical gates
SHA3-384	2^{210} classical gates
AES-256	2^{285}/MAXDEPTH quantum gates or 2^{272} classical gates
SHA3-512	2^{274} classical gates

Table 1.2: Quantum and classical gates required to break AES and SHA3 [24]

The MAXDEPTH parameter signifies the constant running time or circuit depth required for quantum attacks. This parameter serves as a measure of the time and resources that a quantum computer would need to compromise the suggested signature schemes.

The complexity of these quantum attacks can be assessed by considering their circuit size, which can then be compared to the number of gates required by classical computers to perform equivalent computations. Table 1.2, provided by NIST, illustrates the classical and quantum gate counts for optimal key recovery and collision attacks on AES and SHA-3, respectively. The data in the table is derived by limiting circuit depth to MAXDEPTH.

1.4 Pros and Cons of Using Codes

Public key cryptography can be implemented using various techniques. Cryptography built upon error-correcting codes offers specific advantages over other, more popular cryptosystems (such as RSA and ECC). One notable advantage is its relative efficiency. Additionally, code-based cryptography rooted in BD and SD problems boasts provable security. However, the most significant advantage lies in the immunity of code-based cryptosystems to attacks based on quantum computers.

Nevertheless, despite these advantages, code-based cryptosystems have not achieved widespread adoption like RSA. This is largely due to the fact that code-based cryptosystems often entail the use of very large keys. Furthermore, as evident from this report, a secure and efficient code-based signature scheme has not yet been established. As stated earlier, NIST has announced a call for "Additional Digital Signature Candidates for PQC Standardization". In response to this "call for proposals", six code-based signature schemes were submitted in the first round. It is anticipated that this initiative will yield one or more secure and efficient code-based signature schemes in the future.

Chapter 2

A Novel Code-based Signature Scheme

2.1 Introduction

This chapter presents a novel code-based signature scheme. The given signature scheme is very efficient as no decoding is required and only matrix multiplication in binary field is sufficient to generate and verify signature. In addition, the presented scheme does not use any trapdoor and hence any random linear code can be used in the scheme.

The given scheme is based on Kabatianskii-Krouk-Smeets (KKS) type FDH code-based signature scheme [17]. KKS type schemes have several points working in their favor which recommend them to be the basis of a novel code-based signature scheme. The KKS signature scheme does not require decoding during signature generation, which makes the scheme very efficient. Moreover, no trapdoor (i.e. efficiently decodable linear code) is required, which makes setting up the scheme very efficient. However, KKS as given, suffers from a very simple attack as shown by Cayrel, Otmani and Vergnaud [8]. The scheme given in this chapter has been designed to resist Cayrel, Otmani and Vergnaud's attack.

In the following subsection the KKS signature scheme is reviewed. The Cayrel, Otmani, Vergnaud (COV) attack on the scheme is described in subsection 2.1.2. The end of subsection 2.1.2 also discusses the reason that the new scheme is immune to the attack. The scheme with an example is presented in section 2.2. Section 2.3 explains why the scheme is correct.

2.1.1 The Kabatianskii-Krouk-Smeets Signature Scheme

In 1997, G. Kabatianskii, E. Krouk, and B. Smeets introduced a novel signature based on the Niederreiter cryptosystem-like signature scheme (referred to as the KKS scheme) [17]. This marked the first utilization of code-based cryptography in a Full Domain Hash (FDH) scheme. The general KKS signature scheme is outlined as follows:

- **Setup:** Similar to before let H_A be a random parity check matrix of size $(n-k) \times n$

corresponding to linear code $C_A[n, k, \geq d]$. Let V be another $[n', k, d_1]$ linear code with $n' < n$ and generator matrix $G' = [g_{ij}]_{k \times n'}$. The Hamming weight of any code in V be smaller then or equal to d_2. Let J be a subset of $\{1, 2, \ldots, n\}$ of cardinality n'.

The signer computes $k \times n$ matrix G^* such that the columns of G^* belonging to J are equal to the columns of G', and zero otherwise. The signer additionally constructs $H_A(J)$ by selecting n' columns from H_A, each with an index in J. The signer now calculates

$$F = H_A(J) G'^T \tag{2.1}$$

- **Private Keys:** The private keys include the subset $J \subset \{1, 2, \ldots, n\}$ and the $k \times n'$ matrix G'.
- **Public Keys:** The public keys comprise the $(n-k) \times k$ matrix F and the $(n-k) \times n$ parity check matrix H_A.
- **Message Signing:** The signature of a message m is computed as $f(m) = mG^*$. Notably, the length of message m is k, while its signature length is n. Moreover, the signature weight falls between d_1 and d_2.
- **Signature Verification:** The recipient of the message-signature pair (m, s) verifies the signature by confirming that $d_1 \leq wt(s) \leq d_2$ and by validating the equation:

$$Fm^T = H_A s^T \tag{2.2}$$

This equation holds because $H_A G^{*T} = H_A(J) G'^T$.

2.1.2 Cayrel Otmani and Vergnaud Attack on KKS Scheme

Kabatianskii et al.'s signature scheme marked a significant advancement as the first FDH signature scheme that appeared to address all the security vulnerabilities that plagued earlier signature schemes. However, in 2007, Pierre-Louis Cayrel, Ayoub Otmani, and Damien Vergnaud uncovered a rather straightforward key recovery attack on the KKS scheme [8].

Cayrel et al.'s attack hinges on the observation that the zero bits in the signature $f(m)$ reveal the contents of set J. This is due to the construction of matrix G^*, wherein the i_{th} column of G^* takes the form of a $[0]_{k \times 1}$ vector if and only if $i \notin J$. Consequently, if the r_{th} bit of $f(m)$ is non-zero, it is certain that $r \in J$. However, if the r_{th} bit of $f(m)$ is zero, no definitive conclusions can be drawn regarding the membership of r within the set J. Consequently, having access to multiple signatures is likely to unveil the complete set J.

Once the set J is known, one can construct $H_A(J)$, and from there, by solving equation (2.1), derive the value of private key G'.

In the new scheme this attack is no longer applicable due to the way the matrix G^* is constructed. The matrix G^* does not have any all zero column, as its columns are permutations of columns of two generator matrices G_1 and G_2. Thus, the zero (or non-zero) bits of sig does not reveal any information about the structure of secret matrix G^*.

2.2 The Scheme

1. **Key Generation:** Let the parity check matrix of code $C\,[n, k, 2d+1]$ be H_A. Let $C_1\,[n_1, l, d_1]$ and $C_2\,[n_2 = n - n_1, l, d_2]$ be two secret codes having generator matrices G_1 and G_2 and $d_1 + d_2 = d$.

   H_A is the public key and matrices G_1 and G_2 are private keys. A one-way hash function $h : \{0,1\}^* \rightarrow \{0,1\}^l$ is also chosen.

2. **Message Signature:** Let m be the message that need to be signed. Let $J \subset \{1, 2, \ldots, n\}$ be a set of cardinality n_1.

   Construct matrix G^* which has columns of G_1 at indices $i \in J$ and columns of G_2 for the rest of the indices. G^* is a $l \times n$ matrix. Let $F = H_A {G^*}^T$. Matrix F is a $(n-k) \times n$ matrix. A random salt r is chosen and then sig is calculated as

   $$sig = h\,(m \| r)\, G^* \tag{2.3}$$

   The quantity sig is accepted as the signature of message m if $wt\,(sig) = d$. Otherwise, a new r is chosen and the calculation of sig is repeated untill the weight condition is satisfied.

   After successful calculation of sig the tuple (m, r, sig, F).

3. **Verification:** On receiving the message-signature triple (m, r, sig, F), verification is carried out by first checking that $wt\,(sig) = d$, and then,

   $$Fh\,(m \| r)^T = H_A sig^T \tag{2.4}$$

In the given scheme the private/public key pair need to be changed after every l iterations (the reason for this is given in section 3.3.4).

Example 2.2.1 *Let us consider a random linear code $C\,[15, 11]$ whose parity-check matrix is given by*

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Now consider $C_1\,[7,4]$ *code with generator matrix*

$$G_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

and, code $C_2\,[8,4]$ *with generator matrix*

$$G_2 = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Dependent row $v_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$ *is created by adding first and third rows of* H*. Similarly row* $v_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$ *is created by adding the first and last rows of* H*. The rows* v_1 *and* v_2 *are randomly inserted in second and third row positions of* H *and the second and third rows are appended at the end to give rise to*

$$H_A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The matrix H_A *is public and the matrices* G_1 *and* G_2 *are private.*

Let the hash of the message to be signed be $h\,(m) = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$*. A set* $J \subset \{1, 2, \ldots, 15\}$ *of size* 7 *is randomly chosen such that* $J = \{3, 4, 5, 6, 8, 9, 11\}$*. Then we construct the* 4×15 *matrix* G^* *by filling its columns in* J *by the columns of* G_1 *and rest by the columns of* G_2*. Thus, the value of* G^* *is constructed as*

$$G^* = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The public key F is calculated as

$$F = H_A G^{*T}$$

$$= \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then, the signature of the message m is

$$sig = h(m) G^*$$

$$= \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

During verification, the receiver of the message-signature tuple (m, sig, F), first computes

$$Fh\,(m)^T = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

and then calculates

$$H_A sig^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Since, $Fh\,(m)^T = H_A sig^T$, the signature is verified.

2.3 Correctness of the scheme

The given scheme correctly generates verifiable signature for an honest signer. The first thing to note is that $wt\left(h\left(m\|r\right)G^{*}\right)=d$ can be satisfied. This can be seen by noting that the columns of G^{*} are the permutations of columns of G_1 and G_2. Thus, any codeword of G^{*}, which sig is, has weight greater than or equal to $d_1+d_2=d$. Moreover,

$$\begin{aligned} Fh\left(m\right)^{T} &= H_A G^{*T} h\left(m\right)^{T} \\ &= H_A\left(h\left(m\right)G^{*}\right)^{T} \\ &= H_A sig^{T} \end{aligned} \tag{2.5}$$

Thus, any signature generated following the steps outlined in the scheme should be correctly verified by the receiver.

Chapter 3

Analysis of the Novel Signature Scheme

3.1 Introduction

In the previous chapter a novel code-based signature scheme based on Kabatiaskii et. al.'s signature scheme [17] has been described. In this chapter an analysis of the scheme is given. This analysis consists of a preliminary estimation of the cost of running the algorithm (section 3.2). The cost analysis takes into account both time as well as space requirement of the algorithm. Section 3.3 presents an theoretical analysis of the security of the scheme.

The analysis of the signature scheme has been done by assuming the computational syndrome decoding (CSD) problem is hard for a random linear code.

3.2 Theoretical Analysis of the Cost of the Algorithm

In this and all further analysis we assume that we are working in the binary field.

3.2.1 Storage Cost of Public Key

The given digital scheme uses one public code $C\,[n, k, 2d + 1]$ and two private codes $C_1\,[n_1, l, d_1]$ and $C_2\,[n - n_1, l, d_2]$, where $n_1 < n$ and $d_1 + d_2 = d$. The public key is C's parity check matrix H_A. The size of matrices H_A is $(n - k) \times n$.

3.2.2 Storage Cost of Private Key

The private keys are the generator matrices G_1 and G_2 of the codes C_1 and C_2, respectively. The size of G_1 and G_2 are $l \times n_1$ and $l \times (n - n_1)$, respectively. The total storage size of the private keys is $(l \times n)$ bits.

3.2.3 Communication Cost

The signature of the message m is given by the pair $(r, h(m\|r) G^*, F)$. The matrix G^* has dimension of $(l \times n)$. Hence, $h(m\|r) G^*$ has length n and $F = H_A G^{*T}$ is a matrix of size $(n-k) \times l$. Hence, the total size of the signature is $n + (n-k) \times l$ bits. This can be taken as the communication cost of the given scheme.

3.2.4 Theoretical Cost of Signature Generation

The theoretical cost of generating the signature is now calculated as follows. Let C_h be the cost of hashing. Then, the generation of signature $sig = h(m\|r) G^*$, involves one hash operation and a vector-matrix multiplication between vector of size l and matrix of size $l \times n$. The vector-matrix multiplication operation involves maximum $(l \times n)$ xor operations.

In addition, every time a signature is generated, the set J is randomly chosen with cost n_1 and the matrix $F = H_A G^{*T}$ is calculated. The operation of generating F consists of xoring the n columns of H_A of length $(n-k)$ for each column of G^{*T} (which has l columns). Hence, the maximum cost of generating F is $n(n-k)l$. Thus, the total cost of signature generation is upper-bounded by $C_h + n_1 + nl(n-k+1)$ operations.

The components of the signature generation cost is summarised as follows:

1. **Cost of hashing:** C_h
2. **Cost of calculating** sig**:** nl
3. **Cost of generating set** J**:** n_1
4. **Cost of calculating** F**:** $nl(n-k)$
5. **Total Cost:** $C_h + n_1 + nl(n-k+1)$

3.2.5 Theoretical Cost of Signature Verification

The cost of signature verification can be calculated by similar analysis. During signature verification the receiver verifies that $wt(sig) = d$ and then $Fh(m\|r)^T = H_A sig^T$. The cost of weight verification is same as the length of sig which is n. The calculation of $Fh(m\|r)^T$ involves one hash operation of cost C_h and a matrix-vector multiplication of maximum cost $l \times (n-k)$. Similarly, the cost of matrix-vector multiplication $H_A sig^T$ is maximum $n \times (n-k)$. Moreover, the cost of comparing both vectors is equal to the length of the vectors, which is $(n-k)$. Thus, the total cost of signature verification is upper-bounded by $C_h + n + (n-k)(n+l+1)$.

The cost-components of the signature verification is given below. Table 3.1 summarises the storage and operation costs of the signature algorithm.

1. **Cost of hashing:** C_h
2. **Cost of verifying weight of** sig**:** n

Storage Cost of Public Key (Bits)	Storage Cost of Private Key (Bits)	Communication Cost (Bits)	Cost of Signature Generation	Cost of Signature Verification
$n(n-k)$	nl	$n+l(n-k)$	$C_h + n_1$ $+nl(n-k+1)$	$C_h + n$ $+(n-k)(n+l+1)$

Table 3.1: Storage and Operations Cost of the Signature Algorithm

3. **Cost of calculating** $Fh(m)^T$**:** $l \times (n-k)$
4. **Cost of calculating** $H_A sig^T$**:** $n \times (n-k)$
5. **Cost of comparison:** $(n-k)$
6. **Total Cost:** $C_h + n + (n-k)(n+l+1)$

3.3 Theoretical Analysis of the Security of the Algorithm

A digital signature scheme can be attacked through multiple avenues. One option for breaking the given scheme is to forge a signature given the public key. Another approach may be to break the scheme by deriving the private keys from public relations. Security of the scheme for both these approaches is discussed below.

3.3.1 Security Against Forgery

The security of the signature scheme for forgery (i.e. forging signature from public key) is dependent on the hardness of computational syndrome decoding (CSD) problem. Let us assume that some entity A can successfully masquerade as a sender S by generating a valid signature (of proper weight) of any message m while only having access to S's public key H_A. This implies that A can compute sig in polynomial time which satisfies $Fh(m\|r)^T = H_A sig^T$. However, this means that A can solve the CSD problem in polynomial time for matrix H_A. Since, CSD problem is NP-complete, A should not be able to directly generate sig, given access to H_A.

3.3.2 Public-Private Break

As mentioned above, another avenue of attacks on the scheme may try to derive the private key from public relations $sig = h(m)G^*$ and $F = H_A G^{*T}$. However, given a known message signature pair (m, sig), the only way to find matrix G^* such that $sig = mG^*$, is to try all possible $l \times n$ binary matrices one-by-one to find the specific one which satisfies the relation. As there are a total 2^{ln} such matrices, the search has the worst-case cost of $O\left(2^{ln}\right)$.

The other alternative, which is to try to solve the series of equations given by the relation $F = H_A G^{*T}$, is also impractical. Give the $(n-k+\rho) \times l$ matrix F and a $(n-k+\rho) \times n$ matrix H_A we can try to solve for each column of G^{*T}. For each column of G^{*T}, this results in $n-k+\rho$ equations in n variables. As the number of variables is larger than the number of equations, such a system of equation will have $2^{(k-\rho)}$ possible solutions. As there are l columns of G^{*T}, finding the correct G^* takes $O\left(l \cdot 2^{(k-\rho)}\right)$ trials in the worst case.

3.3.3 Attack on J (Choice of Private Parameters)

One important point to keep in mind is the fact that the set J should be chosen independently and randomly for every message that needs to be signed. This is because, if the same J is repeated with multiple messages then partial information about the private key G^* may be revealed. Assuming that the same J is used in the protocol, a fraudster may get a message consisting of zeroes except a one at the i_{th} position to reveal the i_{th} row G^*. Below we derive an expression for the expected number of messages that need to signed using the same key in order to reveal the entire matrix G^* using the above attack.

Set J consist of n_1 elements from the set $\{1, 2, \ldots, n\}$. Let E_i be the random variable representing the number of iterations in which the same J is picked i times. Clearly, E_i is geometrically distributed and having mean value $\binom{n}{n_1}^i$. In order to reveal the full matrix G^*, the same J needs to be picked l times. Thus, on the average we can expect $\binom{n}{n_1}^l$ iterations of the protocol with the same key, before the above mentioned attack can reveal the full matrix G^*.

3.3.4 Known Signature Attack

One avenue of breaking the signature is to collect l number of message-signature pairs (m_i, sig_i) and from those recover the private key G^*. The message-signature pairs give rise to the following matrix equation

$$S = MG^* \tag{3.1}$$

where, S is the $l \times n$ matrix given by

$$S = \begin{bmatrix} sig_1 \\ sig_2 \\ \vdots \\ sig_l \end{bmatrix} \tag{3.2}$$

and, M is the $l \times l$ matrix

$$M = \begin{bmatrix} h(m_1) \\ h(m_2) \\ \vdots \\ h(m_l) \end{bmatrix} \tag{3.3}$$

If the hash of the messages are independent, then M can be inverted and G^* can be derived from equation (3.1).

In order analyze this type of attack we first find the probability that a random collection of l hash values results in an invertible matrix. Since the hash values are pseudorandom, we treat them as random non-zero binary strings of length l.

The probability that the i_{th} row of H is in the span of the first $i-1$ rows is $\left(2^{i-1}-1\right)/2^l$. For H to be invertible every row of H must be independent of all previous rows. Thus the total probability that H is invertible is given by

$$P_H = \prod_{i=1}^{l} \left(1 - \frac{2^{i-1}-1}{2^l}\right) \tag{3.4}$$

For instance, for the parameter value $l = 20$, the probability that H is invertible is approximately 0.3.

In order to prevent this type of attack the scheme should be repeated with the same keys for only l times, and new keys should be chosen after every l rounds.

3.3.5 Signature Forgery Attack

In the given scheme, the signature of an unknown message can be calculated from known message-signature pair (m, sig). Let m' be a message for which we want to find a valid signature sig'. Let

$$M = h\left(m\right) + h\left(m'\right) \tag{3.5}$$

Then,

$$\begin{aligned} Fh\left(m'\right)^T &= F\left(M + h\left(m\right)\right)^T \\ &= FM^T + Fh\left(m\right)^T \\ &= FM^T + H_A sig^T \\ &= H_A\left(RFM^T + sig^T\right) \\ &= H_A\left(MF^TR^T + sig\right)^T \end{aligned} \tag{3.6}$$

Here R is the right-inverse of the matrix H_A. Thus, using (m, sig), an attacker may be able to forge a valid signature for message m' as $sig' = MF^TR^T + sig$.

However, as we show below, it is highly unlikely that $sig' = MF^TR^T + sig$ will meet the requisite weight condition of d, and hence will be rejected during verification.

For the following discussion we assume that the message m' is chosen at random and hence the vector $x = MF^TR^T$ can be assumed to be a random vector of length n. We now calculate the probability that a random vector added with sig gives rise to another vector of weight d.

Let the support of vector x have t common elements with the support of sig. This elements can be chosen in $\binom{d}{t}$ ways. For each of these choices there are $\binom{n-d}{d-t}$ ways to

choose the remaining support positions of x. Thus, there are total $\sum_{t=1}^{d} \binom{d}{t}\binom{n-d}{d-t}$ ways to choose x.

Hence, the probability of getting a x of correct format is

$$P = \frac{\sum_{t=1}^{d} \binom{d}{t}\binom{n-d}{d-t}}{2^n} \tag{3.7}$$

Now we assume that the attacker is choosing random messages one after the other until they get a valid x. This is a random experiment following geometric distribution with pmf given by equation (3.7). Thus, the average number of messages that the attacker must choose before he gets a proper x is $1/P$.

For parameter values $n = 220$ and $d = 22$ the expected number of messages is calculated as 1.88×10^{36}.

3.3.6 Notes on Future Work

Future work should focus on a more detailed analysis of the key selection process for the scheme, particularly how keys will be chosen every l iterations. Additionally, the computational cost of key generation at each l-th iteration must be carefully evaluated. Furthermore, the selection of codes C_1 and C_2 should ensure a sufficient number of minimum-distance codewords to maintain the scheme's viability.

A thorough investigation into these aspects will be included in upcoming reports.

Chapter 4

Performance Analysis of the Signature Scheme for Different Parameters

4.1 Introduction

In this chapter we present the practical performance analysis of the signature scheme. All runtime data given in the chapter has been derived by running a ANSI C-language implementation of the scheme which has been executed in a computer running Ubuntu 24.04.1 LTS Operating System with Intel Core i5 processor and 16 GB of RAM. The sections of this chapter are arranged as follows.

The next section gives an overview of the implementation used in deriving all the run time data. In section 4.3 the running time of the signature implementation is calculated for three set of parameters. This is done for both precomputed and computed keys. In sections 4.4 and 4.5 the plots of running time of the implementation against code length and code dimension respectively are given. Section 4.6 calculates the memory space required by the keys and the signature for different set of parameters.

4.2 Overview of the Implementation

The implementation of the signature scheme used for deriving the test results has been written in ANSI C programming language and compiled with gcc. It uses the Fast Library for Number Theory (FLINT) [14] for performing various matrix operations and libsodium library for generating random numbers and message hash.

The implementation first fixes the parameters, length, dimension and minimum distance, of the two private codes having generator matrices G_1 and G_2. It provides the option of generating these parameters randomly or entering them as user input. Let $[n_1, l, d_1]$ and $[n_2, l, d_2]$ be the parameters for G_1 and G_2 respectively. Next G_1 and G_2 are generated either randomly or taken as user input from file.

Table 4.1: Execution Time without Precomputed Keys

Set	n_1	n_2	k	d_1	d_2	Exec. Time (sec)
1	40	50	15	6	7	0.011799
2	60	68	20	10	12	0.025211
3	100	120	20	10	12	0.045395

After that a random public parity-check matrix H is generated corresponding to a code with parameters $[n_1 + n_2, k, 2(d_1 + d_2) + 1]$. In our implementation the value of the parameter l is chosen to have the same value. The matrix H is generated by using an efficient implementation of Brouwer-Zimmermann algorithm [36], known as algorithm 994, due to Hernando et. al. [15]. Given a random matrix the algorithm 994 checks whether the matrix satisfies the minimum distance criteria. Once matrix H is generated it is converted to H_A by inserting ρ number of dependent rows.

After that a random message is generated (or taken as user input), its hash calculated using BLAKE2 algorithm [5], and then its signature is generated following the procedure outlined in Chapter 2. The implementation then verifies the signature and writes all the relevant outputs in a file.

As will be seen later on, the most time consuming part of the implementation is the Brouwer-Zimmermann algorithm. One way of speeding the implementation is to precompute the keys and store them in a file. In the forthcoming sections we present the running time for different parameters using both precomputed keys and by computing the keys in runtime.

4.3 Execution Time with and without Precomputed Key

In this section the execution time of the implementation is presented for both precomputed key and without precomputed key. Three sets of parameter values are for these calculations. The execution times without precalculating the keys is presented in table 4.1. In table 4.2 the execution time is given with precomputed keys.

The plot of the execution time for three sets of parameters is presented in figure 4.2. From the plot it can be seen that the variation of execution time is much smaller for the non-precomputed keys than for the precomputed keys.

Figure 4.1 gives a bar graph showing the key generation time, signature time and signature verification time for the three sets of parameters in table 4.1. The plot shows that the key generation is the most expensive operation. The signature scheme can be made more efficient by pre-generating keys and storing them in files.

Table 4.2: Execution Time with Precomputed Keys

Set	n_1	n_2	k	d_1	d_2	Exec. Time (sec)
1	40	50	15	6	7	0.00116
2	60	68	20	10	12	0.002064
3	100	120	20	10	12	0.005607

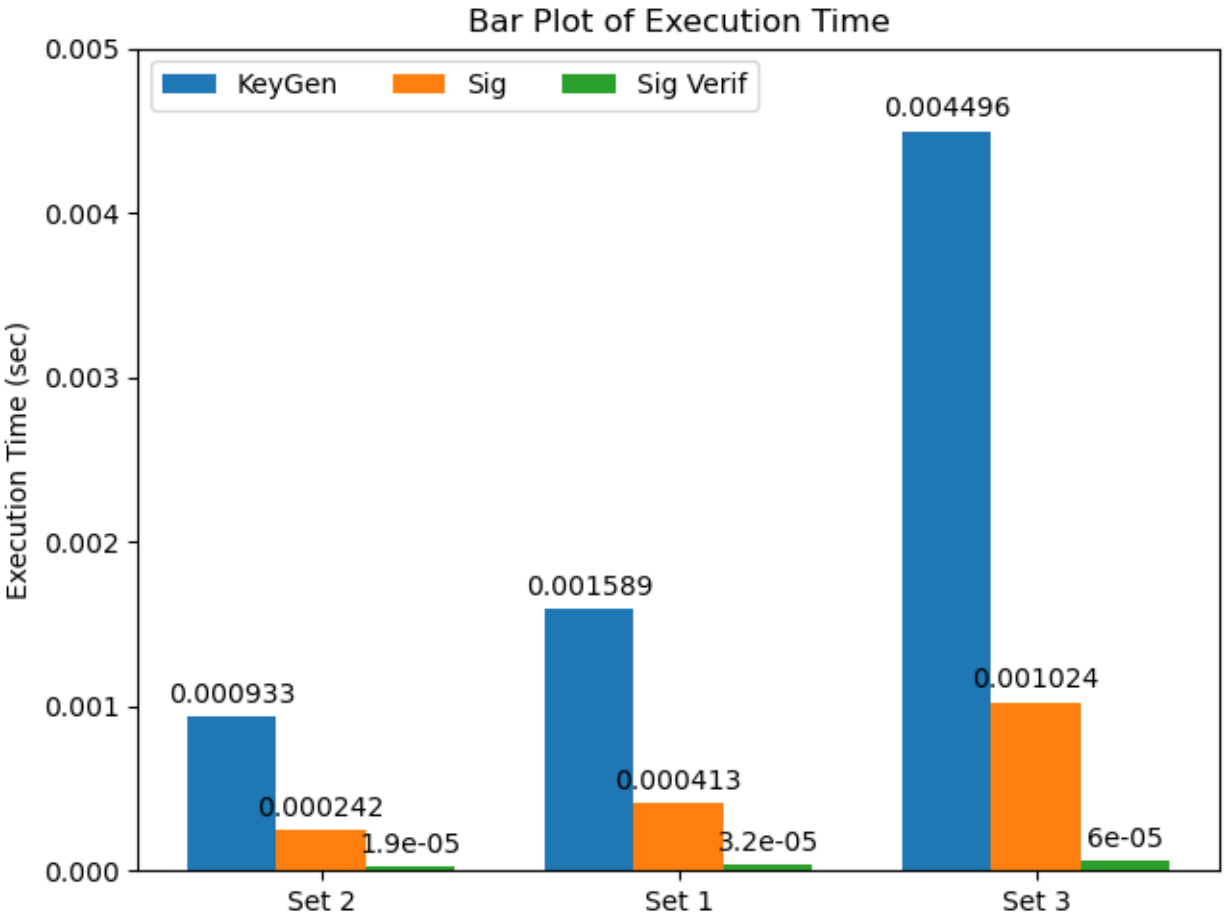

Figure 4.1: Bar Plot Execution Time

4.4 Execution Time versus Code Length

In this section the plot of the execution time is presented while varying the code length parameter n_1, maintaining $n_2 = n_1 + 10$, and fixing $k = 15$ and $d_1 = d_2 = 6$. The parameter n_1 is varied from 40 to 100 in steps of 10. The plot is shown in figure 4.3.

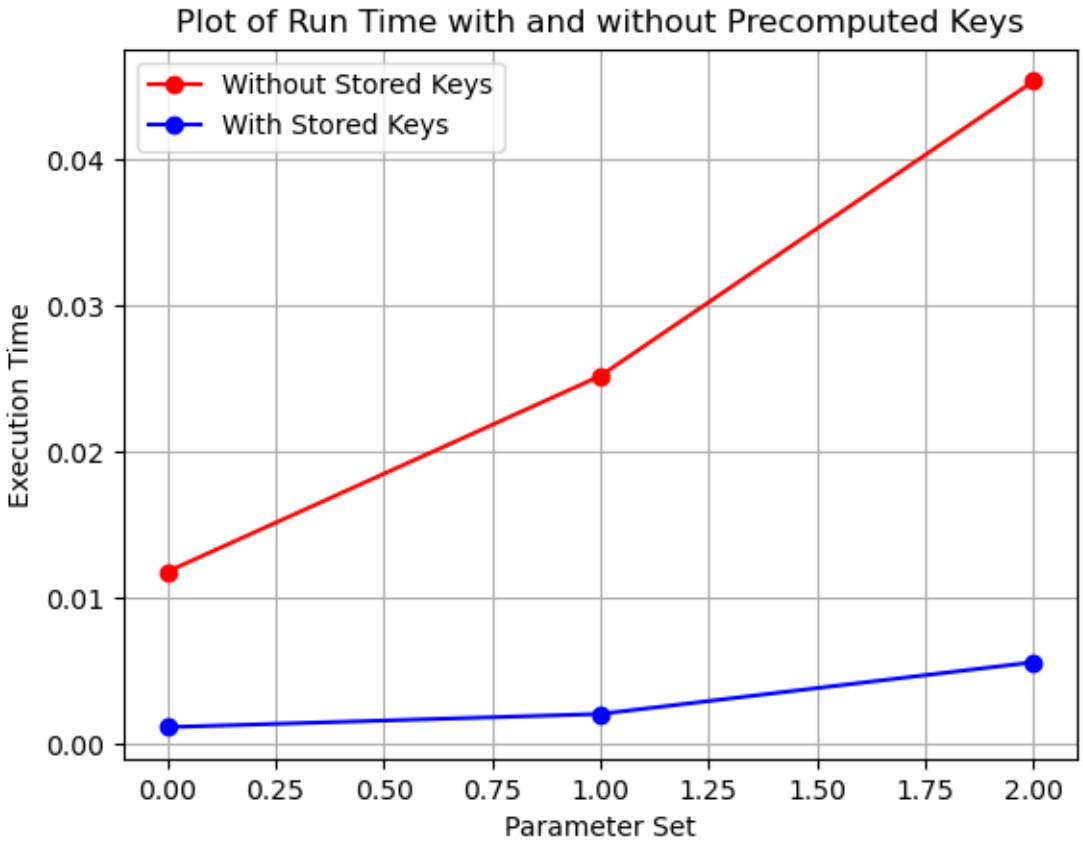

Figure 4.2: Plot of Execution Time for Stored and Unstored Keys

4.5 Execution Time versus Code Dimension

This section gives the plot of execution time for varying code dimension (k). For generating the plot $n_1 = 40$, $n_2 = 50$ and $d_1 = d_2 = 10$ has been set. The parameter k is varied from 10 to 20 in steps of 2. The plot is shown in figure 4.4.

4.6 Keys and Signature Sizes

In the section the key and signature space requirement are given (table 4.3). The in-memory sizes are calculated for three set of parameters given in table 4.1. The space requirements of the matrices and the signature is plotted in figure 4.5.

Figure 4.3: Plot of Execution Time against Code Length

Table 4.3: Size of keys and Signature

Set	G_1 (Kb)	G_2 (Kb)	H_A (Kb)	F (Kb)	Sig (Kb)
1	39.8	49.4	437.25	77.25	6.27
2	78.53	88.77	892.1	145.6	8.7
3	129.72	155.33	2829.25	269.25	14.59

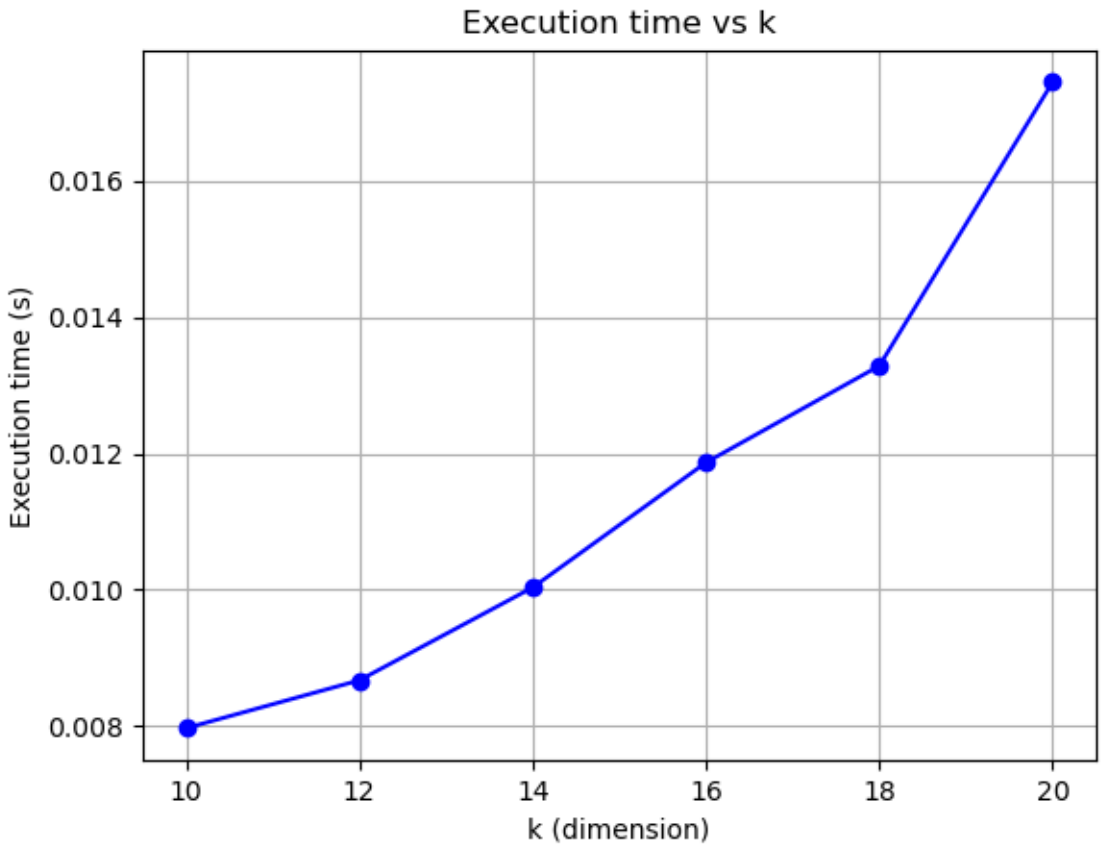

Figure 4.4: Plot of Execution Time against Code Dimension

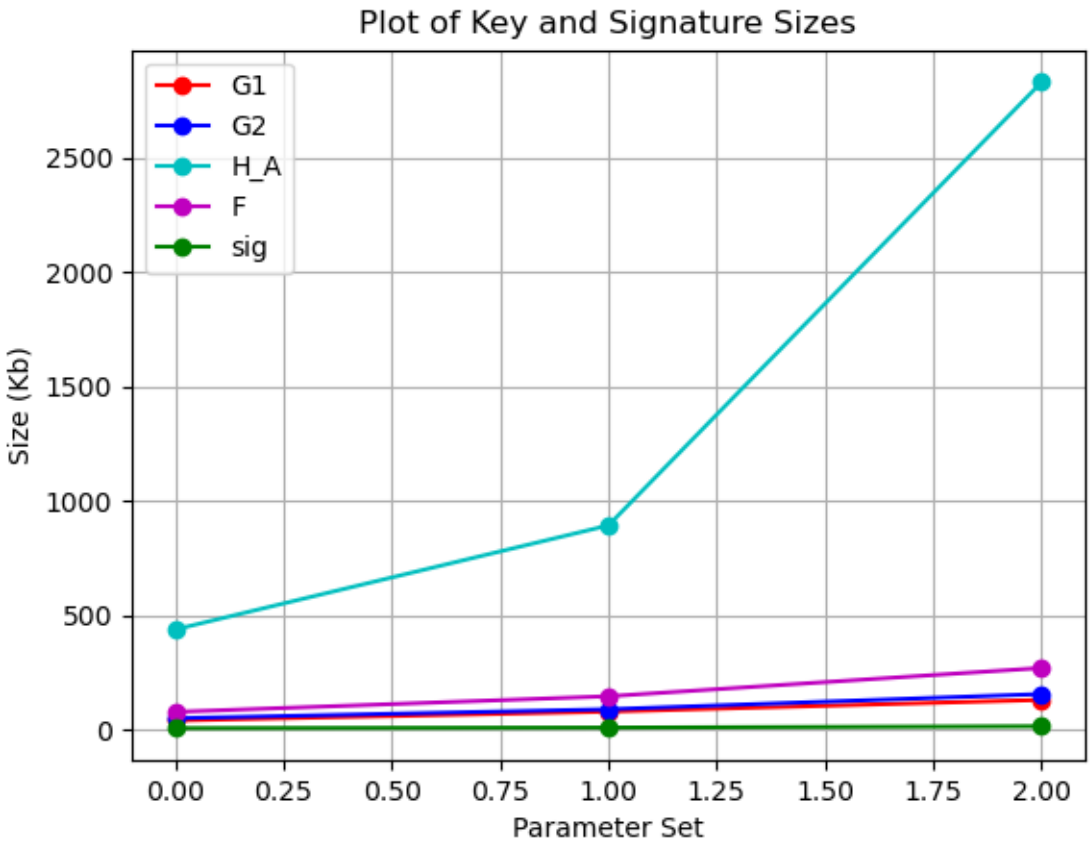

Figure 4.5: Plot of Key and Signature Sizes

Chapter 5

Theoretical Cryptanalysis of Proposed System

5.1 Introduction

In this chapter we present the formal cryptanalysis of the proposed scheme. We first give a formal security reduction of the scheme to syndrome decoding problem (SDP), thus proving that breaking the scheme is equivalent to breaking the SDP. Next we show that the proposed scheme is EUF-CMA secure. Finally we present some sample parameter values of the scheme which attend 128-bits, 192-bits and 260-bits security levels.

5.2 Security of the Proposed Scheme

Like the KKS scheme, the security of the scheme is based on the syndrome decoding problem (SDP).

Claim 1 *Given the public keys the derivation of the private key is analogous to breaking the SDP.*

Proof.
Given the public key H_A and the matrix F, the only way to derive the private key G^* is to solve the relation

$$F = H_A G^{*T} \tag{5.1}$$

Let g_i^* be the i^{th} row of G^*, and f_i be the i^{th} column of F. Then the above equation gives rise to l equations of the form

$$f_i = H_A g_i^{*T} \tag{5.2}$$

Each of the above equation is a syndrome decoding problem, and hence solving (5.1) is equivalent to solving l SDPs. $\diamond$

It can be shown that given the public key and message it is equally hard to derive the signature without knowing the private key.

Claim 2 *Given a random message m (whose signature is not known) and public key H_A and F calculating the signature of m is equivalent to solving the SDP.*

Proof.
In order to forge the signature of a message m, the attacker needs to find a vector s' such that $Fh\left(m\|r\right)^T = H_A s'^T$ and $wt\left(s'\right) = d$. Thus, $Fh\left(m\|r\right)^T$ is the syndrome of s' and hence successfully finding s' is equivalent to solving the SDP. $\diamond$

5.3 EUF-CMA Security

The term EUF-CMA stands for Existential Unforgeability under Chosen Message Attack. A signature is EUF-CMA secure if an attacker can not forge a signature to an unknown message even if the attacker has access to unlimited valid message signature pairs. The attacker is said to have access to a *signing oracle*, from which the attacker can adaptively request valid message-signature pairs. There is no limit to how many message-signature pairs the attacker can request, provided that it is finite.

The EUF-CMA security can be represented as a game played between the attacker and the signing oracle. Let $\mathcal{A}$ be the attacker and $\mathcal{O}$ represent the signing oracle. Then, the EUF-CMA game is played as follows:

- The oracle $\mathcal{O}$ generates keys $(pk, sk) \leftarrow KeyGen\left(\right)$.
- $\mathcal{O}$ shares pk with $\mathcal{A}$.
- $\mathcal{A}$ requests signatures of messages $m_1, m_2, \ldots$ from $\mathcal{O}$.
- $\mathcal{O}$ shares signatures $sig_1, sig_2, \ldots$ with $\mathcal{A}$.
- The above two steps are carried out as many times as $\mathcal{A}$ wants.
- Eventually, $\mathcal{A}$ outputs message-signature pairs (m^*, sig^*).
- If $\mathcal{O}$ has already seen the message m^* in $\mathcal{A}$'s request, then the game is stopped and $\mathcal{O}$ wins.
- Otherwise $\mathcal{A}$ wins if sig^* is successfully verified as m^*'s signature.

We adopt the following strategy to prove that the proposed signature scheme is EUF-CMA secure. We know that SDP is hard [31, 4]. We design a simulator $\mathcal{B}$ and show that if an attacker $\mathcal{A}$ can win the EUF-CMA game then $\mathcal{B}$ can use $\mathcal{A}$ to solve the SD problem. Hence, $\mathcal{A}$ can not win the EUF-CMA game.

Claim 3 *The proposed signature scheme is EUF-CMA secure.*

Proof.

- **Setup:** $\mathcal{B}$ receives the challenge x. $\mathcal{B}$ has to calculate sig^* such that $Fx^T = H_A sig^{*T}$.
- **Simulation:** $\mathcal{B}$ shares the public keys of the scheme with $\mathcal{A}$. $\mathcal{B}$ simulates a random oracle h and keeps a list of queries $h(m_i) = h_i$. It selects a random query index i^* and sets $h(m_{i^*}) = x$.
- **Signing Queries:** When $\mathcal{A}$ asks for signature on message m, $\mathcal{B}$ first checks whether $x = h(m)$, and if so, $\mathcal{B}$ can not answer and aborts. Else, $\mathcal{B}$ returns the signature of m.
- **Forgery:** $\mathcal{A}$ outputs (m^*, sig^*). If $m^* = m_{i^*}$, then $x = h(m^*)$, and $Fx^T = H_A sig^{*T}$. Thus, $\mathcal{B}$ outputs sig^*, and $\mathcal{B}$ has solved SDP.

$\diamond$

5.4 Parameter Sizes of the Modified Scheme for Different Security Levels

In order to estimate the parameter values of the modified scheme we fix the code $\mathbb{C}_1$ and $\mathbb{C}_2$ to be the dual of the binary BCH code. This is done in the original KKS scheme as well. A BCH code (and its dual) is a linear code whose length and minimum distance are determined by two parameters m and t, respectively. For given m and t, the dual of the BCH code is a $[2^m - 1, tm, 2t + 1]$ code.

The parameter m is chosen such that $n_1 = n_2 = 2^m - 1$. The other parameter t is chosen such that the codes $\mathbb{C}_1$ and $\mathbb{C}_2$ have design distance $d_1 = d_2 = 2t + 1$. Being duals of BCH codes, the codes $\mathbb{C}_1$ and $\mathbb{C}_2$ have dimension $l = tm$. For providing 128-bit security the parameters m and t are chosen such that $l \geq 128$.

The length of the code $\mathbb{C}$ is taken to be $n = 2^{m+1} - 2$. The minimum distance of the code $\mathbb{C}$ needs to $2(d_1 + d_2) + 1 = 8t + 5$. The dimension of $\mathbb{C}$, k, is fixed such that it meets the Gibert-Varshamov (GV) bound, i.e., $R = k/n \geq 1 - H_2(\delta)$, where H_2 is the binary entropy function and $\delta = (2(d_1 + d_2) + 1)/n$ is the relative distance of code $\mathbb{C}$. For large enough n (i.e. $n > 100$) this bound is met with very high probability.

For example, for parameters $m = 9$ and $t = 16$, the codes $\mathbb{C}_1$ and $\mathbb{C}_2$ are $[511, 144, 33]$ dual BCH codes. The code $\mathbb{C}$ is $[1022, 450, 133]$ random linear code.

For higher order security the parameter values can be increased accordingly. For example, in order to achieve 192 bit security, the parameters m and t can be assigned the values 10 and 20 respectively. Then we have $\mathbb{C}_1$ and $\mathbb{C}_2$ are dual BCH codes $[1023, 200, 41]$ and code $\mathbb{C}$ is $[2046, 1222, 165]$.

Taking $m = 10$ and $t = 26$, we have $n_1 = n_2 = 2^{10} - 1 = 1023$, $l = 260$ and $d_1 = d_2 = 2t + 1 = 53$. The length of the code $\mathbb{C}$ is $n = 2^{11} - 2 = 2046$ and the minimum distance is $8t + 5 = 213$. The dimension of $\mathbb{C}$ which satisfies the asymptotic GV bond is $k = n\left(1 - H_2\left(\delta\right)\right) \approxeq 1072$.

Thus, for 260 bit security $\mathbb{C}_1$ and $\mathbb{C}_2$ are $[1023, 260, 53]$ codes and $\mathbb{C}$ is $[2046, 1072, 213]$ code.

Codes	**128-bit Security (m = 8, t = 16)**	**192-bit Security (m = 8, t = 24)**	**260-bit Security (m = 10, t = 60)**
C_1 and C_2	$[511, 144, 33]$	$[1023, 200, 41]$	$[1023, 260, 53]$
C	$[1022, 450, 133]$	$[2046, 1222, 165]$	$[2046, 1072, 213]$

Table 5.1: Code Parameters for Different Security Levels

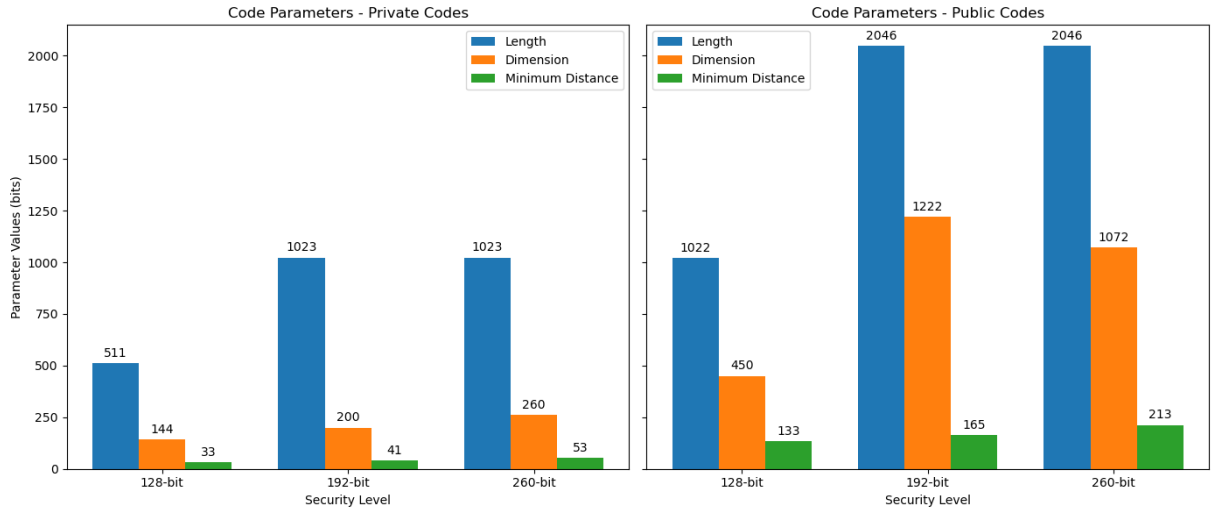

Figure 5.1: Bar charts showing code parameters for different security levels. Private Code and Public Code parameters are shown side-by-side for comparison.

Bibliography

[1] Mohssen Alabbadi and Stephen B Wicker. Security of xinmei digital signature scheme. *Electronics Letters*, 28(9):890–891, 1992.

[2] Mohssen Alabbadi and Stephen B Wicker. Digital signature schemes based on error-correcting codes. In *Proceedings. IEEE International Symposium on Information Theory*, pages 199–199. IEEE, 1993.

[3] Mohssen Alabbadi and Stephen B Wicker. Susceptibility of digital signature schemes based on error-correcting codes to universal forgery. In *Error Control, Cryptology, and Speech Compression: Workshop on Information Protection Moscow, Russia, December 6–9, 1993 Selected Papers*, pages 6–12. Springer, 2005.

[4] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. In *Annual International Cryptology Conference*, pages 92–110. Springer, 2007.

[5] Jean-Philippe Aumasson, Willi Meier, Raphael C-W Phan, Luca Henzen, Jean-Philippe Aumasson, Willi Meier, Raphael C-W Phan, and Luca Henzen. Blake2. *The Hash Function BLAKE*, pages 165–183, 2014.

[6] Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani. Using LDGM codes and sparse syndromes to achieve digital signatures. In *Post-Quantum Cryptography: 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings 5*, pages 1–15. Springer, 2013.

[7] Paulo SL. M. Barreto, Rafael Misoczki, and Marcos A. Simplicio Jr. One-time signature scheme from syndrome decoding over generic error-correcting codes. *Journal of Systems and Software*, 84(2):198–204, 2011.

[8] Pierre-Louis Cayrel, Ayoub Otmani, and Damien Vergnaud. On kabatianskii-krouk-smeets signatures. In *Arithmetic of Finite Fields: First International Workshop, WAIFI 2007, Madrid, Spain, June 21-22, 2007. Proceedings 1*, pages 237–251. Springer, 2007.

[9] Nicolas T. Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology-ASIACRYPT*

2001: 7th International Conference on the Theory and Application of Cryptology and Information Security, volume 7, pages 157–174. Springer, December 2001.

[10] Léonard Dallot. Towards a concrete security proof of Courtois, Finiasz and Sendrier signature scheme. In *Western European Workshop on Research in Cryptology*, pages 65–77. Springer, 2007.

[11] Matthieu Finiasz. New constructs using error-correcting codes in public key cryptography. *Doctoral thesis, École Polytechnique*, 2004.

[12] Danilo Gligoroski, Simona Samardjiska, Håkon Jacobsen, and Sergey Bezzateev. McEliece in the world of Escher. *Cryptology ePrint Archive*, 2014.

[13] L Harn and D. C. Wang. Cryptanalysis and modification of digital signature scheme based on error-correcting code. *Electronics Letters*, 2(28):157–159, 1992.

[14] William Hart, Fredrik Johansson, and Sebastian Pancratz. Fast library for number theory, 2011.

[15] Fernando Hernando, Francisco D Igual, and Gregorio Quintana-Ortí. Algorithm 994: fast implementations of the brouwer-zimmermann algorithm for the computation of the minimum distance of a random linear code. *ACM Transactions on Mathematical Software (TOMS)*, 45(2):1–28, 2019.

[16] Andreas Huelsing, Daniel J. Bernstein, Lorenz Panny, and Tanja Lange. Official NIST Comments made for RaCOSS. Technical report, Official NIST Comments made for RaCOSS, 2018.

[17] Gregory Kabatianskii, Evgenii Krouk, and Ben Smeets. A digital signature scheme based on random error-correcting codes. In *Crytography and Coding: 6th IMA International Conference Cirencester, UK, December 17–19, 1997 Proceedings 6*, pages 161–167. Springer, 1997.

[18] Wijik Lee, Young-Sik Kim, Yong-Woo Lee, and Jong-Seon No. Post quantum signature scheme based on modified Reed-Muller code pqsigRM. Technical report, First round submission to the NIST post-quantum cryptography call, November 2017.

[19] Yuan-Xing Li. An attack on xinmei's digital signature scheme. In *Proceedings. IEEE International Symposium on Information Theory*, pages 236–236. IEEE, 1993.

[20] Yuan Xing Li, Robert H Deng, and Xin Mei Wang. On the equivalence of mceliece's and niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1):271–273, 1994.

[21] R. J. McEliece. A public-key system based on algebraic coding theory, 114-116. deep sace network progress report, 44. *Jet Propulsion Laboratory, California Institute of Technology*, 1978.

[22] Dustin Moody and Ray Perlner. Vulnerabilities of "McEliece in the World of Esche". In *Post-Quantum Cryptography: 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings 7*, pages 104–117. Springer, 2016.

[23] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Prob. Contr. Inform. Theory*, 15(2):157–166, 1986.

[24] NIST. Post-quantum cryptography: Digital signature schemes, 2023. Accessed on Oct 30, 2023.

[25] Ayoub Otmani and Jean-Pierre Tillich. An efficient attack on all concrete KKS proposals. In *Post-Quantum Cryptography: 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29–December 2, 2011. Proceedings 4*, pages 98–116. Springer, 2011.

[26] Ray Perlner, Dustin Moody, and Jacob Alperin-Sheriff. Official NIST comments made for pqsigRM. Technical report, NIST PQC first round submission, 2018.

[27] Aurélie Phesso and Jean-Pierre Tillich. An efficient attack on a code-based signature scheme. In *Post-Quantum Cryptography: 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings 7*, pages 86–103. Springer, 2016.

[28] Partha Sarathi Roy, Kirill Morozov, Kazuhide Fukushima, Shinsaku Kiyomoto, and Tsuyoshi Takagi. Code-based signature scheme without trapdoors. *IEICE technical report*, 118(152):17–22, 2018.

[29] Partha Sarathi Roy, Rui Xu, Kazuhide Fukushima, Shinsaku Kiyomoto, Kirill Morozov, and Tsuyoshi Takagi. RaCOSS (random code-based signature scheme). Technical report, First Round Submission to NIST Post Quantm Cryptography Call, 2017.

[30] Nicolas Sendrier. Public-key cryptosystems based on error-correcting codes, habilitation to supervise research, pierre and marie curie university, paris 6, paris, france. *Mars*, 2002.

[31] HCA Van Tilborg, RJ Mceliece, and ER Berlekamp. On the inherent intractability of certain coding problems. 1978.

[32] Johan van Tilburg. Cryptanalysis of xinmei digital signature scheme. *Electronics Letters*, 28(20):1935–1936, 1992.

[33] Viktoriya Vladimirovna Vysotskaya and Ivan Vladimirovich Chizhov. The security of the code-based signature scheme based on the stern identification protocol. *Prikladnaya diskretnaya matematika*, (57):67–90, 2022.

[34] Keita Xagawa. Practical attack on RaCOSS-r. *Cryptology ePrint Archive*, 2018.

[35] Wang Xinmei. Digital signature scheme based on error-correcting codes. *Electronics Letters*, 26(13):898–899, 1990.

[36] Karl-Heinz Zimmermann. Integral hecke modules, integral generalized reed-muller codes, and linear codes. Technical report, Techn. Univ. Hamburg-Harburg, 1996.