Ingénierie des Systèmes d'Information
Vol. 26, No. 6, December, 2021, pp. 585-590
Journal homepage: http://iieta.org/journals/isi

Towards the Construction of Reed-Muller Code Based Symmetric Key FHE

Ratnakumari Challa[1*], VijayaKumari Gunta[2]

[1] Computer Science and Engineering, RGUKT-RK Valley, IIIT-AP, Kadapa 516330, Andhra Pradesh, India
[2] Computer Science and Engineering, JNTUH, Hyderabad, Telangana 500085, India

Corresponding Author Email: ratnamala3784@gmail.com

https://doi.org/10.18280/isi.260609

Received: 15 October 2021
Accepted: 3 December 2021

Keywords:
reed-muller code, coding theory, erroneous codewords, permutation, majority logic decoding

ABSTRACT

Homomorphic encryption (HE) schemes became popular cryptographic primitives and very useful in variety of security applications. Homomorphic encryption based on coding theory have the advantages of faster computations due to the structural properties of the codes used. Several schemes are supporting unlimited $Mod\ 2$ addition operations in literature. The present paper introduces Reed-Muller (RM) code based $Mod\ 2$ multiplication operation thereby making RM code based HE scheme fully homomorphic. The representation of the codeword with necessary adaption to support unlimited number of $Mod\ 2$ multiplication operations is presented along with the scheme first. The correctness proof of the homomorphic operations along with experimental evaluation is also presented to demonstrate the practical aspects of the proposal.

1. INTRODUCTION

Homomorphic encryption is powerful cryptographic tool that enables secure computations on the private data. It evaluates any function for any operation securely on the encrypted data without knowing its corresponding plaintext. For original data p, c denotes the ciphertext of the original plaintext p, i.e., $c = E_k(p)$. The encryption scheme E_k is said to be homomorphic with respect to some set of operations O, if, for any operation o ∈ O one can compute $E_k(p_1 \text{ o } p_2)$ from $E_k(p_1) \text{ o } E_k(p_2)$.

In 1978, Rivest et al. investigated first design for homomorphic encryption scheme [1] and that was broken in 1987 [2]. Later research was carried on to determine the secure schemes which can perform homomorphic encryption operations efficiently. In 2009, Craig Gentry's breakthrough work [3] has described theoretical construction of Fully Homomorphic encryption scheme. That scheme has become popular model as well as the root for other related developments of FHE schemes. Based on Gentry's breakthrough work, a number of developments, variations and new FHE schemes were proposed. Few of such developments and variants are: the FHE scheme based on worst-case hardness on lattice problems [4]; Gentry's scheme with reduced key and ciphertexts [5]; FHE scheme with practical implementation [6]; FHE based on Learning with Errors over Rings [7]; Ring-LWE based FHE [8]; Homomorphic encryption based on coding theory [9]; levelled FHE without bootstrapping [10]; FHE with fast implementation of operations [11]; and Reed-Muller based FHE using post processing [12]. Integer based HE scheme [13, 14] Homomorphic encryption schemes based on coding theory were developed, they support unlimited addition operations but multiplication operations with some specified depth [9].

Based on the standard theoretical schemes, several implementations of FHE are developed [15]. Few of them are PALISADE [16], HEAAN [17], FV-NFLib [18], NuFHE [19]. FHE with practical time complexities has shown to be very beneficial in secure cloud computing, as well as in a variety of other applications. Despite the fact that FHE can be useful for a wide range of applications, many of them are not currently practicable in terms of FHE due to the schemes' practical constraints in terms of computational complexity and space requirement [15].

In this work, an afford is made to develop FHE based on coding theory with practical time complexities. The operations involved in coding theory are quite simple and straightforward to execute, which make to build an efficient homomorphic system with low computing complexity and good security.

In this paper, we present the symmetric key construction of RM code based Fully Homomorphic Encryption (RMFHE). Reed-Muller codes are error correcting codes. We introduce set of artificial errors at certain positions specified by secret key to compute the ciphertext and use majority logic decoding to recover the message from erroneous codes using secret key. To achieve linear homomorphism, the additive and multiplicative property is preserved over the set.

2. NOTATIONS AND DEFINITIONS

In this section we have presented description of all symbols and notations used in RMFHE scheme. An integer will be expressed as lower case italic letters (for example: n), $\mathbb{F}$ be the arbitrary field. For an integer n, $[n]$ will be denoted as set of integers $[0, 1, 2, \ldots, n-1]$. Vector will be denoted as bold italic lower case letter $\boldsymbol{v} \in \mathbb{F}_2^n$ and matrix will be expressed as sequence of vectors $\mathrm{V} = (\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_k}) \in (\mathbb{F}_2^n)^k$ and it is denoted by uppercase letter. Upper case italic letters will represent the set of some elements x, y and so on, i.e., I = (x, y, …). $\mathcal{F}_{ev}$ denotes any function that evaluates on given input to produce the output. For example, *add* function $\mathcal{F}_{add}$:

$(x, y) \to z$, *i.e.*, $z = \text{addition}(x, y)$. $|\boldsymbol{v}|$ denotes the Hamming weight of the vector $\boldsymbol{v}$, i.e., the number of nonzero elements in the vector, and $supp(\boldsymbol{v})$ denote the positions of the non-zero elements in the vector. The operator $' + '$ denote the addition $Mod\ 2$ operator and $'.'$ denote the multiplication $Mod\ 2$ operator.

2.1 Expanded RM evaluation codes

In this work, our aim is to develop fully homomorphic encryption on the linear codes. The evaluation codes given in Frederik Scheme [9] supports unlimited addition operations over cipher texts but supports limited multiplication operations. To make the scheme fully homomorphic encryption scheme, the actual RM codes are represented in expanded form with slight modification to the codeword representation. Hence, it is called Expanded RM evaluation codes.

Definition 1. RM codeword matrix:
If there is a k - dimensional linear subspace $\mathcal{C}$ of $\mathbb{F}_2^n$, which has a minimum hamming distance of d, then for $(k < n)$, $[n, k, d]$ represents a linear code. In the construction of RMFHE scheme, codeword and erroneous codewords are not exactly taken as given in def. 1 of Frederik's scheme [9]. RM code $\mathcal{C}$ is a set of codeword matrices $\{W_1, W_2, W_3, \dots\}$ where $W_i \in (\mathbb{F}_2^n)^k$ is defined as $\boldsymbol{m}_i \times G_{rm}$, where $\boldsymbol{m}_i \in \mathbb{F}_2^k$ is the message and $G_{rm} \in (\mathbb{F}_2^n)^k$ is a generator matrix for given Reed-Muller input parameters (r, m) for $i = 1,2,3, \dots$.

To hold true for additive property over $\mathcal{C}$, for codewords $W_1, W_2 \in \mathcal{C}$, $W_1 + W_2 \in \mathcal{C}$. We call the codeword matrix $W \in \mathcal{C}$ is an error free codeword and $W \in (\mathbb{F}_2^n)^k \backslash \mathcal{C}$ are erroneous codeword. Erroneous codewords are defined as $W + E$ where $E \in (\mathbb{F}_2^n)^k \backslash 0$ called as error matrix. Error matrix $E \in (\mathbb{F}_2^n)^k$ can be taken as set of error vectors $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \dots\}$ where $\boldsymbol{e}_i \in \mathbb{F}_2^n$ for all $i = 1,2,3, \dots$. Each Erroneous codewords has some good locations and bad locations. The bad locations are the positions of error occurrences, and are specified by $supp(\boldsymbol{e})$ and remaining error free positions are known as good locations, i.e., $[n] \backslash supp(\boldsymbol{e})$. For a subset $I \subset [n]$, we define $\mathcal{C}(I) = \{W + E | W \in \mathcal{C}$ and $E \in (\mathbb{F}_2^n)^k$ for each vector $\boldsymbol{e} \in E, supp(\boldsymbol{e}) \subseteq [n] \backslash I\}$. This is called set of erroneous codeword matrices.

Definition 2. Computing original RM codeword from expanded codeword:
We define a step to compute the RM codeword $\boldsymbol{w} \in \mathbb{F}_2^n$ from the codeword matrix $W \in \mathcal{C}$. As W is sequence of vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \dots, \boldsymbol{w}_k$; $W = (\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \dots, \boldsymbol{w}_k) \in (\mathbb{F}_2^n)^k$ then adding of all vectors will result the RM codeword corresponding to RM codeword matrix i.e., $\boldsymbol{w} = \boldsymbol{w}_1 + \boldsymbol{w}_2 + \boldsymbol{w}_3 + \dots + \boldsymbol{w}_k$. These codewords are now similar to the original RM codeword where we can directly apply the decoding algorithm to recover its original plaintext. Decoding of plaintext message from erroneous codewords using majority logic is now similar to the Frederik's scheme [9].

Definition 3: Permutation function:
We define a function $\sigma_S: (\mathbb{F}_2^n)^k \to (\mathbb{F}_2^n)^k$ for permuting the elements of matrix $V \in (\mathbb{F}_2^n)^k$ specified by the permutation key S where $S \in \left(\mathbb{F}_{(x,y)}^n\right)^k$ is denoted as matrix of shuffled indices of (x, y). For example, the permutation on vector V_1 results to V_2. That is given as $V_2 \leftarrow \sigma_S(V_1)$ where $V_1, V_2 \in (\mathbb{F}_2^n)^k$ and $V_2(i, j) = V_1(i', j')$ for all indices $i = 0,1,2, \dots, k-1$ and $j = 0,1,2, \dots, n-1$ and $(i', j') = S[i, j]$.

Definition 4: Addition and multiplication operations:
We define the addition $(+)$ and multiplication $(.)$ operations on the vector/matrix as the component wise $Mod\ 2$ addition and $Mod\ 2$ multiplication respectively over finite fields GF(2).

3. IDEA OF THE PROPOSED SCHEME

An abstract level description of the RMFHE scheme and the design principles of scheme are presented. The proposed scheme constructs Fully Homomorphic encryption scheme based on coding theory. Several works [20-24] have studied and presented on construction of encryption schemes based on coding theory. As depicted in the Figure 1, the basic idea is to get a codeword $\boldsymbol{w} \in \mathbb{F}_2^n$ using encoding technique for a given plaintext $\boldsymbol{m} \in \mathbb{F}_2^k$. Then embedding artificial errors into codeword $\boldsymbol{w}$ to get the ciphertext $\boldsymbol{c}$. Using the secret key, the erroneous positions are identified which makes the decryption easy. However, decryption for an attacker with unknown error positions (secret key) is supposedly as hard as decoding a random code.

The proposed scheme is an extension and variant of the scheme proposed by Kiayias et al. [20, 25, 26] and Armknecht et al. [9]. Encryption works in three steps: (i) at first, encoding over a plaintext produces a codeword (error-free) and (ii) then, some random errors at fixed positions, given by the secret key, are embedded in the codeword to get an erroneous codeword (iii) then, apply a fixed permutation on the erroneous codeword specified by a secret pattern to provide additional security. Similar to the scheme presented elsewhere [9], proposed scheme makes use of the concept of linear codes and it also maintains codeword synchronization.

Due to symmetric key, the erroneous positions as well as good positions repeat for each encryption, and component-wise operations over the ciphertexts also preserves the good positions and thus satisfies the homomorphic property.

The proposed FHE scheme is designed based on the Reed-Muller coding. Reed-Muller code (RM Code) is $[n, k, d]$ linear code and popular error correcting code. According to Reed-Muller coding [27-29], every message $\boldsymbol{m} \in \mathbb{F}_2^k$ is mapped to codeword $\boldsymbol{w} \in \mathbb{F}_2^n$ i.e. $\boldsymbol{w}$=*encode*($\boldsymbol{m}$) using $\boldsymbol{G_{rm}} \in (\mathbb{F}_2^n)^k$, where $\boldsymbol{G_{rm}}$ is a generator matrix of size $k \times n$. RM code has capability to auto correct the errors at the time of decoding when the number of errors in the received codeword is less than $d/2$. Decoding using majority logic recover the message $\boldsymbol{m}$ from the received erroneous codeword $\boldsymbol{w}'$ i.e., $\boldsymbol{m}$ = *decode* ($\boldsymbol{w}'$) when number of errors in $\boldsymbol{w}'$ is less than $d/2$. Several researches [26, 30] stated that RM codes are best suitable as cryptographic primitives in security systems. The RMFHE scheme in principle follows the idea of adding artificial errors at the fixed position in the codeword specified by secret key (between $d/2$ to d positions) yield to give an erroneous codeword $\boldsymbol{w}'$ as ciphertext. Decoding of this erroneous codeword using secret key permit to produce the plaintext message uniquely. When an addition operation is applied on two ciphertexts, decoding of the resultant ciphertext again permit to produce plaintext which is equal to the result of addition on the original plaintexts. That means it holds true for additively homomorphic property. But this approach does not necessarily satisfy multiplicatively homomorphic encryption. Hence, to get multiplicative homomorphic encryption, we consider the RM codeword in expanded form (matrix form) and is denoted as $\mathrm{W} \in (\mathbb{F}_2^n)^k$ instead of codeword vector $\boldsymbol{w} \in \mathbb{F}_2^n$. The proposed RMFHE with encryption and decryption is presented in the Figure 2.

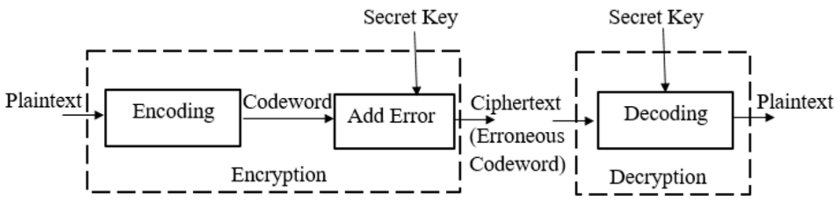
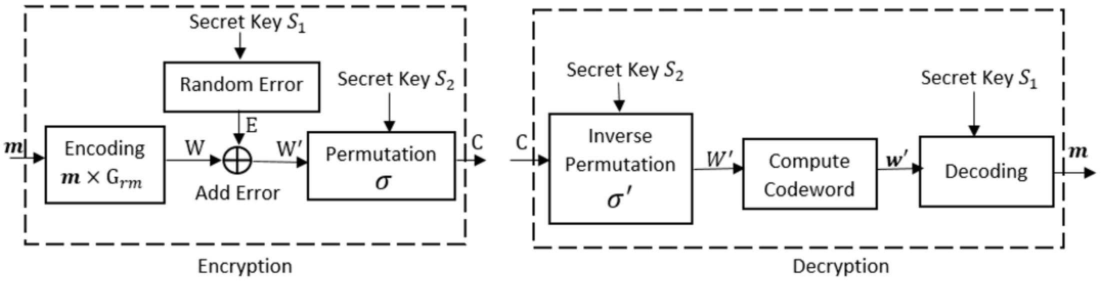


Figure 1. Basic idea of coding theory based encryption



m-Plaintext; *G*$_{rm}$-Generator Matrix; *W*-Codeword matrix; *W′*-Erroneous codeword matrix; *E*-Error Matrix; *C*-Ciphertext; *w′*-Erroneous codeword

Figure 2. Proposed RMFHE scheme

4. RMFHE SCHEME

Present section provides a formal description of constructing the RMFHE scheme. The scheme is symmetric key fully homomorphic construction. This scheme is composed of four functions- Setup, Encryption, Decryption and Evaluation. Each function of the scheme is described is as follows:

Choose the initial RM parameters r, m where $m \geq 2$ and $0 < r \leq m$.

<u>Setup</u>: $K \leftarrow Setup(r, m)$

Setup function take the initial RM parameters (r, m) as input and compute the secretkey $K = (S_1, S_2)$ as an output. The setup algorithm chooses a key S_1 for representing the error positions in the codeword and another key S_2 as permutation key depending on the other parameters; length of the codeword n, dimension k and hamming distance d of RM code.

The algorithm for Setup is given as follows:

1. Given r, m , the RM code computes

$$k = 1 + \begin{bmatrix} m \\ 1 \end{bmatrix} + \begin{bmatrix} m \\ 2 \end{bmatrix} + \begin{bmatrix} m \\ 3 \end{bmatrix} + \cdots + \begin{bmatrix} m \\ 3 \end{bmatrix}$$
$$n = 2^m \text{ and } d = 2^{m-1}$$

2. Choose $S_1 \subset [n]$, set of error locations of length between $d/2$ and d.
3. Choose the permutation key S_2 of $k \times n$ matrix.
4. Output the secretkey $K = (S_1, S_2)$.

<u>Encryption</u>: $\mathrm{C} \leftarrow Encrypt(K, \boldsymbol{m})$

An encryption function $\mathcal{F}_{Encrypt}: \mathbb{F}_2^k \times K \to (\mathbb{F}_2^n)^k$ gets a plaintext $\boldsymbol{m} \in \mathbb{F}_2^k$ and a secretkey $K = (S_1, S_2)$ as inputs. Encryption function transforms the plaintext $\boldsymbol{m}$ into a ciphertext matrix $\mathrm{C} \in (\mathbb{F}_2^n)^k$. First, it encodes the plaintext $\boldsymbol{m}$ into a codeword matrix $\mathrm{W} \in \mathcal{C}$ using generator matrix G_{rm}. Then it samples a random error matrix $\mathrm{E}_{S_1} \in (\mathbb{F}_2^n)^k \backslash 0$ for each vector $\boldsymbol{e} \in \mathrm{E}_{S_1}, supp(\boldsymbol{e}) \subseteq S_1$ to produce the erroneous codeword matrix $\mathrm{W}' = \mathrm{W} + \mathrm{E}_{S_1}$ where $\mathrm{W}' \in \mathcal{C}(S_1)$. Finally, the permutation function σ_{S_2} is applied on W' to get the ciphertext C using the permutation key S_2. The ciphertext matrix C is the output of the encryption function. Given a message plaintext vector $\boldsymbol{m} \in \mathbb{F}_2^k$ as an input to encryption algorithm, it computes the ciphertext C as follows:

$$C = \sigma_{S_2}(\boldsymbol{m} \times G_{rm} + E_{S_1})$$

<u>Decryption</u>: $\boldsymbol{m} \leftarrow Decrypt(K, \mathrm{C})$

The inputs to decryption function $\mathcal{F}_{Decrypt}: (\mathbb{F}_2^n)^k \times K \to \mathbb{F}_2^k$ are ciphertext matrix $\mathrm{C} \in (\mathbb{F}_2^n)^k$ and secret key $K = (S_1, S_2)$. Decryption function first computes the erroneous codeword matrix $\mathrm{W}' \in (\mathbb{F}_2^n)^k$ using inverse permutation function $\sigma'$with permutation key S_2. Then, it computes the RM codeword $\boldsymbol{w}' \in \mathbb{F}_2^n$ from the erroneous codeword matrix W' by adding all its rows as given in Definition. 2 of Section 2. Since the codeword matrix is erroneous, the computed codeword is also erroneous. Then the erroneous codeword $\boldsymbol{w}'$ can be decoded to recover the plaintext $\boldsymbol{m} \in \mathbb{F}_2^k$ by using majority logic decoding with known error positions specified by S_1. Given a ciphertext matrix C as an input to the decryption algorithm, it computes the plaintext vector $\boldsymbol{m}$ as follows:

1. Compute Codeword matrix W' from C using inverse permutation function σ'_{S_2}

$$\mathrm{W}' = \sigma'_{S_2}(\mathrm{C})$$

2. Compute the RM codeword $\boldsymbol{w}' \in \mathbb{F}_2^n$ from codeword matrix W'

3. Extract the plaintext vector $\boldsymbol{m}$ from the codeword $\boldsymbol{w}'$ using majority logic decoding function

$$\boldsymbol{m} = Decode(S_1, \boldsymbol{w}')$$

<u>Addition</u>: $\mathrm{C}_3 \leftarrow Add(\mathrm{C}_1, \mathrm{C}_2)$

Addition function $\mathcal{F}_{Add}: (\mathbb{F}_2^n)^k + (\mathbb{F}_2^n)^k \rightarrow (\mathbb{F}_2^n)^k$ gets two ciphertexts $\mathrm{C}_1, \mathrm{C}_2 \in (\mathbb{F}_2^n)^k$ as input where C_1 and C_2 are encryption of plaintext vectors $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$. An encryption of sum of corresponding underlying plaintexts $(\boldsymbol{m_1} + \boldsymbol{m_2})$, as a ciphertext $\mathrm{C}_3 \in (\mathbb{F}_2^n)^k$ is generated. The addition operation on the two ciphertexts is given as follows:

$$\mathrm{C}_3 = \{\mathrm{C}_1 + \mathrm{C}_2 | \mathrm{C}_1(i,j) + \mathrm{C}_2(i,j)\ \forall i = 0,1,2,\dots,k-1 \\ \forall j = 0,1,2,\dots,n-1\}$$

<u>Multiplication</u>: $\mathrm{C}_3 \leftarrow Mul(\mathrm{C}_1, \mathrm{C}_2)$

Multiplication function $\mathcal{F}_{Mul}: (\mathbb{F}_2^n)^k . (\mathbb{F}_2^n)^k \rightarrow (\mathbb{F}_2^n)^k$ gets two ciphertexts $\mathrm{C}_1, \mathrm{C}_2 \in (\mathbb{F}_2^n)^k$ as input and computes the product of corresponding underlying plaintexts as a ciphertext $\mathrm{C}_3 \in (\mathbb{F}_2^n)^k$. The multiplication operation on the two ciphers texts is given as follows:

$$\mathrm{C}_3 = \{\mathrm{C}_1 . \mathrm{C}_2 | \mathrm{C}_1(i,j) . \mathrm{C}_2(i,j)\ \forall i = 0,1,2,\dots,k-1 \\ \forall j = 0,1,2,\dots,n-1\}$$

5. CORRECTNESS PROOF OF THE PROPOSED RMFHE SCHEME

Proposition 1:

The scheme is correct for encryption, decryption, addition and multiplication operations.

Theorem 1:

The scheme proposed above is correct with respect to fresh cipher text as well as homomorphic operations addition $Mod\ 2$ and multiplication $Mod\ 2$ over finite fields GF(2) as defined. (Bitwise AND and XOR Gentry circuit based unlimited operations supported).

Proof:

Decryption is implemented simply based on majority logic based RM decoder. When errors are within the limit between $d/2$ and d, decoding works correct only with the known error positions.

Let C be the ciphertext corresponding to the plaintext vector $\boldsymbol{m}$. Decryption algorithm deciphers the plaintext $\boldsymbol{m}$ by using two step technique to eliminate the masks on the plaintext. In the first step, it removes the permutation mask using secretkey S_2. In the second step, it extracts the plaintext vector $\boldsymbol{m}$ using RM error correction decoding with known error positions in the erroneous codeword specified by secretkey S_1. Deciphering process by eliminating the masks is illustrated as follows

$$\begin{aligned} \boldsymbol{m} &= Decrypt(S_1, \sigma'_{S_2}(\mathrm{C})) \\ &=> Decrypt(S_1, \sigma'_{S_2}(\sigma_{S_2}(\boldsymbol{m} \times \mathrm{G}_{rm} + \mathrm{E}_{S_1}))) \qquad (1) \\ &=> Decrypt(S_1, \sigma'_{S_2}(\sigma_{S_2}(\mathrm{W} + \mathrm{E}_{S_1}))) \end{aligned}$$

where, W=$\boldsymbol{m}$×G_{rm}.

Inverse permutation operation using key S_2 will remove the permutation mask and then Eq. (1) can now be

$$=> Decrypt(S_1, \mathrm{W} + \mathrm{E}_{S_1})$$

The majority logic decoding of erroneous codeword using secret key S_1 will eliminate the error mask and extract the plaintext vector $\boldsymbol{m}$ as output. Hence, the correctness of decryption of fresh ciphertext is proved. Similarly, correctness of decryption can be proved for addition and multiplication operations over ciphertexts as given below:

Let C_1 and C_2 be the ciphertexts corresponding to two plaintexts $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$. The homomorphic addition on $\mathrm{C}_1$and C_2 will produce C_3 i.e., $\mathrm{C}_3 = \mathrm{C}_1 + \mathrm{C}_2$. The decryption of the resultant cipher deciphers the sum of two plaintext vectors. The correctness of homomorphic addition operation is proved as follows:

$$\begin{gathered} \mathrm{C}_3 = \mathrm{C}_1 + \mathrm{C}_2 \\ => \sigma_{S_2}(\boldsymbol{m_1} \times \mathrm{G}_{rm} + \mathrm{E}_{S_1}) + \sigma_{S_2}(\boldsymbol{m_2} \times \mathrm{G}_{rm} + \mathrm{E}_{S_1}) \\ => \sigma_{S_2}(\boldsymbol{m_1} \times \mathrm{G}_{rm} + \mathrm{E}_{S_1} + \boldsymbol{m_2} \times \mathrm{G}_{rm} + \mathrm{E}_{S_1}) \\ => \sigma_{S_2}(\boldsymbol{m_1} \times \mathrm{G}_{rm} + \boldsymbol{m_2} \times \mathrm{G}_{rm} + \mathrm{E}_{S_1} + \mathrm{E}_{S_1}) \end{gathered}$$

Since $\mathrm{E}_{S_1} + \mathrm{E}_{S_1}$ will generate another error matrix $\mathrm{E}'_{S_1}$with random bits at the positions specified by S_1 and remaining bits will become 0s.

$$\begin{gathered} => \sigma_{S_2}((\boldsymbol{m_1} + \boldsymbol{m_2}) \times \mathrm{G}_{rm} + \mathrm{E}'_{S_1}) \\ \mathrm{C}_3 = \sigma_{S_2}((\boldsymbol{m_1} + \boldsymbol{m_2}) \times \mathrm{G}_{rm} + \mathrm{E}'_{S_1} \end{gathered}$$

Decryption of the result cipher C_3 will result $\boldsymbol{m_3} = \boldsymbol{m_1} + \boldsymbol{m_2}$. Hence, homomorphic addition operation is proved.

Finally, for homomorphic multiplication operation.

Let C_1 and C_2 be the ciphertexts corresponding to two plaintexts $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$. The homomorphic multiplication on $\mathrm{C}_1$and C_2 will produce C_3 i.e., $\mathrm{C}_3 = \mathrm{C}_1 . \mathrm{C}_2$. The decryption of the resultant cipher deciphers the product of two plaintext vectors. The correctness of homomorphic multiplication operation is proved as follows:

$$\begin{gathered} \mathrm{C}_3 = \mathrm{C}_1 . \mathrm{C}_2 \\ => \sigma_{S_2}(\boldsymbol{m_1} \times \mathrm{G}_{rm} + \mathrm{E}_{S_1}) . \sigma_{S_2}(\boldsymbol{m_2} \times \mathrm{G}_{rm} + \mathrm{E}_{S_1}) \\ => \sigma_{S_2}(\mathrm{W}_1 + \mathrm{E}_{S_1}) . \sigma_{S_2}(\mathrm{W}_2 + \mathrm{E}_{S_1}) \\ => \sigma_{S_2}((\mathrm{W}_1 + \mathrm{E}_{S_1}) . (\mathrm{W}_2 + \mathrm{E}_{S_1})) \\ => \sigma_{S_2}(\mathrm{W}_1 . \mathrm{W}_2 + \mathrm{W}_1 . \mathrm{E}_{S_1} + \mathrm{W}_2 . \mathrm{E}_{S_1} + \mathrm{E}_{S_1} . \mathrm{E}_{S_1}) \\ => \sigma_{S_2}(\mathrm{W}_1 . \mathrm{W}_2 + \mathrm{E}'_{S_1}) \end{gathered}$$

Since $\mathrm{W}_1 . \mathrm{E}_{S_1} + \mathrm{W}_2 . \mathrm{E}_{S_1} + \mathrm{E}_{S_1} . \mathrm{E}_{S_1}$ will generate another error matrix E'_{S_1} with random bits at the positions specified by S_1 and remaining bits will become 0s.

$$\begin{gathered} => \sigma_{S_2}((\boldsymbol{m_1} . \boldsymbol{m_2}) \times \mathrm{G}_{rm} + \mathrm{E}'_{S_1}) \\ \mathrm{C}_3 = \sigma_{S_2}((\boldsymbol{m_1} . \boldsymbol{m_2}) \times \mathrm{G}_{rm} + \mathrm{E}'_{S_1}) \end{gathered}$$

Decryption of the result cipher C_3 will result $\boldsymbol{m_3} = \boldsymbol{m_1} . \boldsymbol{m_2}$. Hence, homomorphic multiplication operation is proved.

6. EXPERIMENTAL EVALUATION

The proposed RM FHE scheme is implemented in JAVA platform to execute all the functions of the scheme: Encryption, Decryption, Homomorphic operations. The program was run on the system with basic configurations: Intel(R) Core(TM) i5-3230M CPU 2.60 GHz and 4GB RAM, in Windows 10 Professional 64-bit Operating system environment. The experimental evaluation is conducted to measure the practical time required for every function of the proposed scheme for different parameter values. The implementation of the scheme for various levels of security is obtained. The values of the various parameters corresponding to the different security levels (Toy, small, Medium and Large) is explored and presented in the Table 1. The time required for the functions and operations at different security levels is also obtained and presented in Table 2. The space required to store the secret key (key1 and key2) and ciphertext is presented in Table 3.

The execution results shown that the scheme is quite simple for implementation. Given novelty of the scheme, we consider the results are quite promising at the different security levels. Comparative to the other coding theory based homomorphic schemes (Fredrik's Scheme [9]), the scheme supports unlimited addition and multiplication operations over ciphertext with reasonable time effort. The time for decryption and size of the ciphertext is high compared to other similar schemes. However, it is considered that the decryption is required only at the end when the user decrypt ciphertext after performing all required homomorphic operations by the third party. Hence, there is no much effect of the decryption time on the efficiency of the scheme. Over all the practical implementation of our scheme is possible for FHE with high efficiency, compared to the other schemes based on Lattice cryptography [16], CKKS [17]. RLWE [18].

Table 1. Parameter values at different levels

Level of Security	(r,m)	Codeword length n	Codeword dimension k	Minimum distance d	Auto error correcting capability d_{min}
Toy	(1,3)	8	4	4	1
Small	(1,5)	32	6	16	7
Medium	(1,8)	256	8	128	63
Large	(1,11)	2048	12	1024	511
Very Large	(1,15)	32768	16	16384	8191

Table 2. Practical performance of the scheme at different security levels

	Encryption (in Sec)	Decryption (in Sec)	Addition (in Sec)	Multiplication (in Sec)
Toy	0.000019110	0.00010892	0.00000382	0.00000299
Small	0.00016293	0.00203898	0.00001671	0.00001939
Medium	0.000204300	0.26720770	0.000216690	0.000172310
Large	0.010767670	4.822113740	0.000069260	0.000147250
Very Large	6.046015519	36879	0.000889150	0.003025590

Table 3. Space requirement of Key and Ciphertext at different security level

	Size of S_1 (in KBytes)	Size of S_2 (in KBytes)	Length of C (in bits)	Size of C (in KBytes)
Toy	< 1	< 1	32	< 1
Small	< 1	< 1	192	< 1
Medium	< 1	17	2304	< 1
Large	14	75	24576	16
Very Large	34	1662	5,24,288	98

7. CONCLUSIONS

In this paper an efficient variation of Fredrik's scheme [9] is presented with correctness proof and experimental evaluation. To achieve multiplicative homomorphism, Reed-Muller code has been represented and considered in an expanded form (matrix form), i.e., $(\mathbb{F}_2^n)^k$ instead of $\mathbb{F}_2^n$. The scheme supports unlimited Mod 2 addition and multiplication operations over ciphertexts without refreshing operation. The experimental evaluation of the scheme is performed at different levels of security and the results shown that implementation of our scheme is really practical with high efficiency as compared to other practical implementations of FHE schemes such as schemes based on RLWE, Lattice cryptography, CKKS, in terms of simple and easy operations involved in the scheme.

REFERENCES

[1] Rivest, R., Adleman, L., Dertouzos, M. (1978). On data banks and privacy homomorphisms. Foundations of Secure Communication, pp. 169-177.

[2] Brickell, E.F., Yacobi, Y. (1987). On privacy homomorphisms. Advances in Cryptology – Proceedings of EUROCRYPT'87, the Netherlands, pp. 117-125. https://doi.org/10.1007/3-540-39118-5_12

[3] Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC'09), New York, NY, USA, pp. 169-178. https://doi.org/10.1145/1536414.1536440

[4] Gentry, C. (2010). Toward basing fully homomorphic encryption on worst-case hardness. Advances in Cryptology- Proceedings of CRYPTO'10, Santa Barbara, CA, USA, pp. 116-137. https://doi.org/10.1007/978-3-642-14623-7_7

[5] Smart, N.P., Vercauteren, F. (2010). Fully homomorphic encryption with relatively small key and ciphertext sizes. Public Key Cryptography (PKC'10), Paris, France, pp. 420-443. https://doi.org/10.1007/978-3-642-13013-7_25

[6] Gentry, C., Halevi, S. (2011). Implementing gentry's fully-homomorphic encryption scheme. Advances in Cryptology – EUROCRYPT 2011, Tallinn, Estonia, pp.

129-148. https://doi.org/10.1007/978-3-642-20465-4_9

[7] Lyubashevsky, V., Peikert, C., Regev, O. (2010). On ideal lattices and learning with errors over rings. Advances in Cryptology – EUROCRYPT 2010, French Riviera, pp. 1-23. https://doi.org/10.1007/978-3-642-13190-5_1

[8] Brakerski, Z., Vaikuntanathan, V. (2011). Fully homomorphic encryption from ring-LWE and security for key dependent messages. Advances in Cryptology-CRYPTO'11, Santa Barbara, CA, USA, pp. 505-524. https://doi.org/10.1007/978-3-642-22792-9_29

[9] Armknecht, F., Augot, D., Perret, L., Sadeghi, A.R. (2011). On constructing homomorphic encryption schemes from coding theory. Cryptography and Coding, Oxford, UK, pp. 23-40. https://doi.org/10.1007/978-3-642-25516-8_3

[10] Brakerski, Z., Gentry, C., Vaikuntanathan, V. (2014). (Leveled) fully homomorphic encryption without bootstrapping. Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, New York, USA, pp. 309-325. https://doi.org/10.1145/2633600

[11] Ducas, L., Micciancio, D. (2015). FHEW: Bootstrapping homomorphic encryption in less than a second. Advances in Cryptology -- EUROCRYPT 2015, Sofia, Bulgaria, pp. 617-640. https://doi.org/10.1007/978-3-662-46800-5_24

[12] Challa R., Gunta V. (2016). Reed-muller code based symmetric key fully homomorphic encryption scheme. Information Systems Security. Information Systems Security, Jaipur, India, 499-508. https://doi.org/10.1007/978-3-319-49806-5_29

[13] Van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. Advances in Cryptology – EUROCRYPT 2010, French Riviera, pp. 24-43. https://doi.org/10.1007/978-3-642-13190-5_2

[14] Ramaiah, Y.G., Kumari, G.V. (2012). Towards practical homomorphic encryption with efficient public key generation. International Journal on Network Security, 3(4): 10.

[15] Viand, A., Patrick, J., Anwar, H. (2021). SoK: Fully homomorphic encryption compilers. arXiv preprint arXiv:2101.07078.

[16] Polyakov, Y., Rohloff, K., Ryan, G.W. (2019). PALISADE Lattice Cryptography Library User Manual (v1.6.0), Tech. Rep., Sep. 2019. Available: https://palisade-crypto.org/documentation.

[17] Cheon, J.H., Kim, A., Kim, M., Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. Advances in Cryptology-ASIACRYPT 2017, Hong Kong, China, pp. 409-437. https://doi.org/10.1007/978-3-319-70694-8_15

[18] Crypto Experts. (2019). FV-NFLlib. GitHub. https://github.com/CryptoExperts/FV-NFLlib, accessed on 1 November 2021.

[19] NuCypher. (2019). A GPU implementation of fully homomorphic encryption on torus. GitHub. https://github.com/nucypher/nufhe, accessed on 1 November 2021.

[20] Kiayias, A., Yung, M. (2007). Cryptographic hardness based on the decoding of Reed-Solomon codes. Cryptology ePrint Archive, Report 2007/153, 2007. https://eprint.iacr.org/2007/153.pdf.

[21] McEliece, R.J. (1978). A public-key system based on algebraic coding theory. The Deep Space Network Progress Report, 114-116.

[22] Niederreiter, H. (1985). A public-key cryptosystem based on shift register sequences. Advances in Cryptology - EUROCRYPT'85, Linz, Austria, pp. 35-39. https://doi.org/10.1007/3-540-39805-8_4

[23] Courtois, N.T., Finiasz, M., Sendrier, N. (2001). How to achieve a McEliece-based digital signature scheme. Advances in Cryptology - ASIACRYPT 2001, Australia, pp. 157-174. https://doi.org/10.1007/3-540-45682-1_10

[24] Baldi, M., Bodrato, M., Chiaraluce, G.F. (2008). A new analysis of the McEliece cryptosystem based on QC-LDPC codes. Security and Cryptography for Networks, Amalfi, Italy, pp. 246-262. https://doi.org/10.1007/978-3-540-85855-3_17

[25] Kiayias, A., Yung, M. (2002). Cryptographic hardness based on the decoding of Reed-Solomon codes. Automata, Languages and Programming, Málaga, Spain, pp. 232-243. https://doi.org/10.1007/3-540-45465-9_21

[26] Katz, J., Lindell, Y. (2007). Introduction to Modern Cryptography, 1st edition, CRC Press.

[27] Macwilliams F.J., Sloane, N.J.A. (1983). The Theory of Error-Correcting Codes. North-Holland Mathematical Library. North Holland.

[28] Abbe, E., Shpilka, A., Wigderson, A. (2015). Reed-Muller codes for random erasures and errors. IEEE Transactions on Information Theory, 61(10): 5229-5252. https://doi.org/10.1109/TIT.2015.2462817

[29] Høholdt, T., Van Lint, J.H., Pellikaan, R. (1998). Algebraic geometry codes. Handbook of Coding Theory, 1(Part 1): 871-961.

[30] Gueye, C.T., Mboup, E.H.M. (2013). Secure cryptographic scheme based on modified Reed Muller codes. International Journal of Security and Its Applications, 7(3): 55-64.