Construction of Reed-Solomon Erasure Codes with Four Parities Based on Systematic Vandermonde Matrices

Leilei Yu and Yunghsiang S. Han, *Fellow, IEEE*

Abstract—In 2021, Tang et al. proposed an improved construction of Reed-Solomon (RS) erasure codes with four parity symbols to accelerate the computation of Reed-Muller (RM) transform-based RS algorithm. The idea is to change the original Vandermonde parity-check matrix into a systematic Vandermonde parity-check matrix. However, the construction relies on a computer search and requires that the size of the information vector of RS codes does not exceed 52. This paper improves its idea and proposes a purely algebraic construction. The proposed method has a more explicit construction, a wider range of codeword lengths, and competitive encoding/erasure decoding computational complexity.

Index Terms—Reed-Solomon code, Reed-Muller transform, storage erasure code, fast algorithm.

1 Introduction

Reed-Solomon (RS) codes are a well-known class of maximum distance separable (MDS) codes. With better storage efficiency than replication schemes, they have been used in many prevalent storage systems, such as Redundant Array of Independent Disks (RAID) [1], Google ColossusFS [2], Facebook HDFS [3], Swift [4], Ceph [5] and so on. In recent years, with the development of distributed technologies, a large number of erasure codes based on RS codes or utilizing generalization of RS codes have been developed, such as locally recoverable codes [6], regenerating codes [7], and partial-MDS codes [8]. They are used to meet different requirements of distributed storage systems. In distributed computing systems, RS codes as polynomial codes can also be explored to mitigate the impact of stragglers [9] or guarantee privacy, resiliency, and security [10]. As an implementation basis of all the applications mentioned above, this paper focuses on improving the computational efficiency of RS codes.

An $(N+T, N)$ RS code can encode an information vector of size N into a codeword of $N+T$ symbols, such that N information symbols can be decoded (recovered) from any N symbols in the codeword. If all information symbols always appear precisely in the codeword, then such code is called a systematic $(N+T, N)$ RS code, and the T additional symbols are called parity symbols. This paper focuses on the systematic RS codes and defaults the mentioned decoding to erasure decoding. In particular, RS codes are constructed over finite fields with complicated algebraic operations, so conventional implementations cannot meet low latency needs. Studies have shown that if each parity symbol is calculated independently, the complexity lower bound is $T - \frac{T}{N}$ XORs per data bit in encoding [11]. However, this bound is not true anymore when the intermediate results of calculating parity symbols can be reused. In 2017, [2] proved that for MDS codes with two and three parity symbols, the encoding that can share intermediate results requires at least two XORs per bit. In addition, [2] proposed an algorithm that asymptotically achieves the complexity of two XORs per bit. Subsequently, [12] proposed an algorithm with an asymptotic complexity of three XORs per bit for RS codes with the number of parity symbols between four and seven. It can be seen from the above that even with the addition of up to four parity symbols, the asymptotic complexity of RS algorithm requires only one extra XOR per bit. In 2023, [13] generalized the work in [12] to support the RS codes with any number of parity symbols, and obtained the RS algorithm with the asymptotic complexity of $\lfloor \lg T \rfloor + 1$ XORs per data bit, where $\lg$ denotes the binary logarithm. The above algorithms are fast computations based on Reed-Muller (RM) transform [13], providing the lowest known asymptotic complexities of RS codes when the information length N approaches infinity.

In practical applications, especially storage systems, the RS codes with small T are often used. For instance, RAID-6 uses the RS code with $T = 2$ to ensure data reliability [1], Google ColossusFS and Facebook HDFS use $(9, 6)$ RS code [2] and $(14, 10)$ RS code [3] to tolerate three and four failed nodes, respectively. The RS codes with four parity symbols are considered in this paper. In 2021, based on [12], [14] presented an improved RM-based algorithm for the RS codes with four parity symbols. The main idea is to find a systematic Vandermonde parity-check matrix composed of an identity matrix and a Vandermonde matrix to replace the original parity-check matrix of RS codes. Thanks to the presence of the identity matrix, the algorithm proposed in [14] eliminates the operation of solving linear equations in encoding and reduces the input size of all RM transforms. However, the method in [14] relies on a

• *L. Yu and Y. S. Han are with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China. E-mail: yuleilei@uestc.edu.cn, yunghsiangh@gmail.com.* *This work was supported by the National Key Research and Development Program of China under Grant 2022YFA1004902.(Corresponding author: Yunghsiang S. Han.)*

computer search and does not explicitly give the final result of the searched parity-check matrices. Furthermore, the size N of the information vector in [14] cannot exceed 52. This motivates us to present in this paper a clear and regular result from the perspective of algebraic construction. The main contributions of this paper are listed as follows:

1) From a purely algebraic perspective, we construct a systematic Vandermonde parity-check matrix for the RS codes with four parity symbols.
2) Based on the constructed parity-check matrix, we propose a new RM-based RS encoding/erasure decoding algorithm.
3) We show that the proposed algorithms are competitive with other alternative RM-based RS algorithms regarding the number of operations. In addition, the maximum codeword length of the proposed algorithms for fast execution on $\mathbb{F}_{2^M}$ is $2^M + 4$, where M is any power of two. This is the codeword length of four symbols more than the extended RS code over $\mathbb{F}_{2^M}$ [15].

It is worth noting that the RS algorithm applicable to any number of parity symbols in [13] is also an improvement of the RM-based algorithm in [12]. The difference between this algorithm and the algorithms in this paper and [14] is that it still uses the conventional Vandermonde matrix as a parity-check matrix. This leads to a larger input size for the RM transform required in [13]. In addition, the algorithm in [13] has no improvement on RS decoding but only on encoding. One can see from Sec. 5 that when $T = 4$ both encoding and decoding proposed in this paper are superior to those in [13]. The rest of this paper is organized as follows: Sec. 2 introduces the necessary knowledge of this paper. Sec. 3 and Sec. 4 propose constructing the systematic Vandermonde matrix and the corresponding RS algorithms, respectively. Analysis and comparison are given in Sec. 5. Finally, Sec. 6 concludes this paper.

2 Preliminaries

In this paper, the set of whole numbers is denoted by $\mathbb{N}$. In addition, the decoding mentioned in this paper refers to erasure decoding rather than error-correction decoding [15].

2.1 Finite fields

The finite fields used in this paper come from the field tower presented by Cantor in [16], that is,

$$\begin{aligned}\mathbb{F}_{2^2} :=& \mathbb{F}_2[u_0]/(u_0^2+u_0+1),\\ \mathbb{F}_{2^{2^{i+1}}} :=& \mathbb{F}_{2^{2^i}}[u_i]/(u_i^2+u_i+(u_{i-1}\cdots u_1u_0)),\end{aligned} \tag{1}$$

where $i \in \mathbb{N}\setminus\{0\}$ and $u_0, u_1, u_2, \cdots$ are a specific sequence of elements from an algebraic closure of the binary field $\mathbb{F}_2$. Unless otherwise specified, we suppose that $M = 2^m$, where m is a positive integer. The field tower above leads to $\mathbb{F}_{2^M} = \mathbb{F}_2(u_0, u_1, \cdots, u_{m-1})$. Furthermore, the basis of $\mathbb{F}_{2^M}$ can be denoted by $\boldsymbol{v}_M = (v_0, v_1, \cdots, v_{M-1})$, where $v_i = u_0^{i_0}u_1^{i_1}\cdots u_{m-1}^{i_{m-1}}$ and $(i_{m-1}\cdots i_1 i_0)$ is the binary representation of i, i.e., $i = \sum_{j=0}^{m-1} i_j \cdot 2^j, \forall i_j \in \{0,1\}$. For instance, the basis of $\mathbb{F}_{2^8}$ is $\boldsymbol{v}_8 = (1, u_0, u_1, u_0u_1, u_2, u_0u_2, u_1u_2, u_0u_1u_2)$. From (1), it is not difficult to see that Lemma 1 holds.

Lemma 1. *For any positive integer m, $0 = u_m^2 + u_m + v_{2^m-1}$.*

Since the characteristics of the finite fields above are all two, addition and subtraction are equivalent operations, and Lemma 2 always holds according to [17].

Lemma 2. *For any $i \in \mathbb{N}$ and any two elements a, b that from finite fields in* (1)*, we have that $(a+b)^{2^i} = a^{2^i} + b^{2^i}$.*

In addition, we have the following lemma.

Lemma 3. *Multiplying $a \in \mathbb{F}_{2^{2M}}$ by $b \in \mathbb{F}_{2^M}$ only requires two multiplications over $\mathbb{F}_{2^M}$.*

Proof. From (1), let $a = a_0 + a_1 \cdot u_m$, where a_0 and a_1 are both in $\mathbb{F}_{2^M}$. Then $a \cdot b = (a_0 \cdot b) + (a_1 \cdot b) \cdot u_m$. Since the operations of multiplying a_0 and a_1 with b are performed in $\mathbb{F}_{2^M}$, this lemma holds. □

2.2 Reed-Solomon codes and RM-based algorithm

RS codes can be typically constructed from a Vandermonde parity-check matrix [12]. Formally, let H be a Vandermonde parity-check matrix of $(N+4, N)$ RS code and $\mathbf{r} = (\mathbf{p}|\mathbf{d})$ a codeword of the RS code, where $\mathbf{d}$ is an N-element information vector, and $\mathbf{p}$ is the corresponding 4-element parity vector. All matrices and vectors are over the same finite field, and $N+4$ does not exceed the field size. Hence, the identity $\mathbf{0}^{\mathrm{T}} = H \cdot \mathbf{r}^{\mathrm{T}}$ always holds [12], where $\mathbf{0}$ is a zero row vector. In particular, the identity leads to the following encoding/decoding:

Step C.1: Let $\mathbf{r}'$ be the vector after setting four parity/erased symbols in $\mathbf{r}$ to zero, then calculate $\mathbf{s}^{\mathrm{T}} \triangleq H \cdot (\mathbf{r}')^{\mathrm{T}}$.

Step C.2: Solve the linear system $\mathbf{s}^{\mathrm{T}} = H_e \cdot \mathbf{e}^{\mathrm{T}}$, where $\mathbf{e}$ denote the vector consisting of all parity/erased symbols and H_e denote the sub-matrix of H corresponding to $\mathbf{e}$ (it is clear that $\mathbf{e} = \mathbf{p}$ in encoding).

In 2020, [12] proposed an RM-based algorithm to quickly calculate the following matrix-vector multiplication

$$Vand_4(w_0, w_1, \cdots, w_{N+3}) \cdot \mathbf{x}^{\mathrm{T}}, \tag{2}$$

where $\mathbf{x}$ is a row vector, each $w_i = \sum_{j\in\mathbb{N}} i_j \cdot v_j$ with $i = \sum_{j\in\mathbb{N}} i_j \cdot 2^j$, and $Vand_4(w_0, w_1, \cdots, w_{N+3})$ is a Vandermonde matrix defined by

$$Vand_4(w_0, w_1, \cdots, w_{N+3}) \triangleq \begin{pmatrix} 1 & 1 & \cdots & 1 \\ w_0 & w_1 & \cdots & w_{N+3} \\ w_0^2 & w_1^2 & \cdots & w_{N+3}^2 \\ w_0^3 & w_1^3 & \cdots & w_{N+3}^3 \end{pmatrix}. \tag{3}$$

By setting $H = Vand_4(w_0, w_1, \cdots, w_{N+3})$, it is easy to see that Step C.1 can be efficiently completed by the RM-based algorithm presented in [12]. However, [12] does not develop an efficient calculation for Step C.2.

In 2021, [14] proposed an algorithm that eliminates the operation caused by Step C.2 to improve the performance of the RM-based RS algorithm in [12]. The key idea in [14] is to set an identity matrix on the leftmost side of H, such that H_e in Step C.2 is always an identity matrix that does not produce any operation during encoding. Note that the idea makes some 4×4 sub-matrices of H potentially non-invertible, which results in the decoding not guaranteeing the recovery of any four erased symbols. To solve this

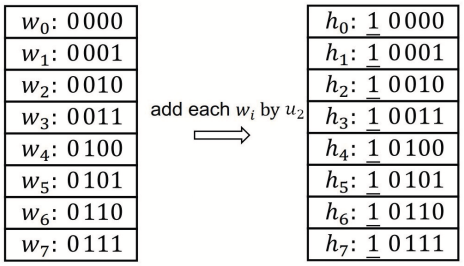

Fig. 1: Construction of the elements in $\{h_i\}_{0\le i<N}$ when $N=8$ and $M=4$.

problem, [14] used computer search to aid in mathematical construction and found some H where any 4×4 sub-matrix is invertible under the new setting, and H is suitable for the RM-based RS algorithm. However, the maximum value of N in [14] does not exceed 52, which was obtained by considering all possible constructions of $\mathbb{F}_{2^8}$.

3 Proposed Construction

Following the idea in [14] but with a different method, this section constructs a systematic Vandermonde parity-check matrix in a purely algebraic manner. In our method, the size N of the information vector can be of any value depending only on the finite field. For simplicity, we assume that N is a power of two.

To begin with, N distinct elements are selected from $\mathbb{F}_{2^{2M}}$, where $N=2^n\le 2^M$ and $n\in\mathbb{N}$. We specify that all selected elements exactly form a set $\{h_i\}_{0\le i<N}$ of each

$$h_i = w_i + u_m \in \mathbb{F}_{2^{2M}}, \tag{4}$$

where the definition of each w_i is the same as that in (3). Furthermore, in the above formula, $w_i\in\mathbb{F}_{2^M}$ and $u_m\in\mathbb{F}_{2^{2M}}\setminus\mathbb{F}_{2^M}$. Fig. 1 shows the setting of $\{h_i\}_{0\le i<N}$ in an example of $N=8$ and $M=4$, where 0-1 sequence denotes the binary representation of the corresponding field element. For example, the sequence 10110 denotes $h_6=v_4+v_2+v_1$, where $v_4=u_2$.

Next, let a parity-check matrix of $(N+4,N)$ RS code be

$$H=(I_{4\times 4}|Vand_4(h_0,h_1,\cdots,h_{N-1}))\,, \tag{5}$$

where $I_{4\times 4}$ is an identity matrix of dimension 4×4. The following is devoted to proving H satisfies the constraint that any 4×4 submatrix is invertible.

According to [14], any 4×4 submatrix of H is invertible as long as the following three submatrices have full rank:

$$\begin{aligned} M_0&=\begin{pmatrix} 1 & 1 & 1\\ h_{i_0}^2 & h_{i_1}^2 & h_{i_2}^2\\ h_{i_0}^3 & h_{i_1}^3 & h_{i_2}^3\end{pmatrix},\\ M_1&=\begin{pmatrix} 1 & 1 & 1\\ h_{i_0} & h_{i_1} & h_{i_2}\\ h_{i_0}^3 & h_{i_1}^3 & h_{i_2}^3\end{pmatrix},\\ M_2&=\begin{pmatrix} 1 & 1\\ h_{i_0}^3 & h_{i_1}^3\end{pmatrix}, \end{aligned} \tag{6}$$

where $i_0,i_1,i_2\in\{0,1,\cdots,N-1\}$ are pairwise distinct. By utilizing generalized Vandermonde determinants [18], it can be computed that the determinants of the matrices above are

$$\begin{aligned} |M_0|&=(h_{i_0}h_{i_1}+h_{i_0}h_{i_2}+h_{i_1}h_{i_2})\cdot\Pi_{0\le j_0<j_1\le 2}(h_{i_{j_0}}+h_{i_{j_1}}),\\ |M_1|&=(h_{i_0}+h_{i_1}+h_{i_2})\cdot\Pi_{0\le j_0<j_1\le 2}(h_{i_{j_0}}+h_{i_{j_1}}),\\ |M_2|&=h_{i_0}^3+h_{i_1}^3. \end{aligned}$$

Since $h_i+h_j\neq 0$ with $i\neq j$, one only needs to check if the following three values are zeros.

1) $h_{i_0}h_{i_1}+h_{i_0}h_{i_2}+h_{i_1}h_{i_2}$: This value equals

$$\begin{aligned} &\sum_{0\le j_0<j_1\le 2}(w_{i_{j_0}}+u_m)(w_{i_{j_1}}+u_m)\\ =&\sum_{0\le j_0<j_1\le 2}\left(w_{i_{j_0}}w_{i_{j_1}}+(w_{i_{j_0}}+w_{i_{j_1}})u_m+u_m^2\right)\\ =&u_m^2+\sum_{0\le j_0<j_1\le 2}w_{i_{j_0}}w_{i_{j_1}}\\ =&u_m+\underbrace{\left(v_{M-1}+\sum_{0\le j_0<j_1\le 2}w_{i_{j_0}}w_{i_{j_1}}\right)}_{a\in\mathbb{F}_{2^M}}, \end{aligned} \tag{7}$$

   where we use the identity $u_m^2=u_m+v_{M-1}$ that comes from Lemma 1. In (7), a is in $\mathbb{F}_{2^M}$ and u_m is in $\mathbb{F}_{2^{2M}}\setminus\mathbb{F}_{2^M}$, thus the value is not zero.
2) $h_{i_0}+h_{i_1}+h_{i_2}$: The value has the form of $b+u_m$, where $b\in\mathbb{F}_{2^M}$. Thus, the value is also not zero.
3) $h_{i_0}^3+h_{i_1}^3$: This value is equal to

$$\begin{aligned} &(w_{i_0}+u_m)^3+(w_{i_1}+u_m)^3\\ &=(w_{i_0}^3+w_{i_1}^3)+(w_{i_0}^2+w_{i_1}^2)\cdot u_m+(w_{i_0}+w_{i_1})\cdot u_m^2\\ &=(w_{i_0}+w_{i_1})\cdot\\ &\left((w_{i_0}^2+w_{i_1}^2+w_{i_0}w_{i_1})+(w_{i_0}+w_{i_1})\cdot u_m+u_m^2\right)\\ &=(w_{i_0}+w_{i_1})\cdot\\ &(\underbrace{(w_{i_0}^2+w_{i_1}^2+w_{i_0}w_{i_1}+v_{M-1})}_{\in\mathbb{F}_{2^M}}+\underbrace{(w_{i_0}+w_{i_1}+1)}_{\in\mathbb{F}_{2^M}}\cdot u_m). \end{aligned} \tag{8}$$

   It is zero only when $w_{i_0}+w_{i_1}+1=0$ and $w_{i_0}^2+w_{i_1}^2+w_{i_0}w_{i_1}+v_{M-1}=0$ at the same time. Equivalently, the conditions can be converted into $w_{i_0}+w_{i_1}=1$ and $w_{i_0}w_{i_1}=v_{M-1}+1$. Assuming the conditions hold, then w_{i_0} and w_{i_1} are two distinct roots of polynomial $f(x)=x^2+x+v_{M-1}+1$. However, the two distinct roots of $f(x)$ are $u_m+u_0\in\mathbb{F}_{2^{2M}}\setminus\mathbb{F}_{2^M}$ and $u_m+u_0+1\in\mathbb{F}_{2^{2M}}\setminus\mathbb{F}_{2^M}$. This is contradictory. So this value is not zero.

From the above, we conclude that any 4×4 submatrix of H is invertible. When $M=8$, the maximum feasible N is $2^M=256$. Note that H in the proposed construction is over $\mathbb{F}_{2^{2M}}$, but the maximum feasible N is equal to the size of $\mathbb{F}_{2^M}$. Since the implementation efficiency of algebraic operation over $\mathbb{F}_{2^{2M}}$ is much slower than that over $\mathbb{F}_{2^M}$, we next propose an RS encoding/decoding algorithm that can be efficiently performed in $\mathbb{F}_{2^M}$.

4 Fast Encoding/Decoding Algorithm

This section presents a fast RS encoding/decoding algorithm for the proposed construction.

4.1 Encoding

According to the encoding process described in Sec. 2.2, let $\mathbf{d} = (d_0, \cdots, d_{N-1}) \in \mathbb{F}_{2^{2M}}^N$ be an information vector and $\mathbf{p} = (p_0, p_1, p_2, p_3) \in \mathbb{F}_{2^{2M}}^4$ the corresponding parity vector. It can be seen from (5) that the formula for calculating all parity symbols is

$$p_\ell = \sum_{i=0}^{N-1} d_i \cdot h_i^\ell, \quad \ell = 0, 1, 2, 3. \tag{9}$$

For convenience, we first calculate

$$\widehat{\mathbf{p}}^{\mathrm{T}} = Vand_4(w_0, \cdots, w_{N-1}) \cdot \mathbf{d}^{\mathrm{T}}, \tag{10}$$

where $\widehat{\mathbf{p}} = (\widehat{p}_0, \cdots, \widehat{p}_3)$. In (10), the calculation of $\widehat{\mathbf{p}}$ can be completed by calling the RM-based algorithm proposed in [12], as shown in (2). Importantly, Lemma 3 implies that the above calculation can be simply performed in $\mathbb{F}_{2^M}$, since each $\widehat{p}_\ell = \sum_{i=0}^{N-1} d_i \cdot w_i^\ell$, where $d_i \in \mathbb{F}_{2^{2M}}$ and $w_i \in \mathbb{F}_{2^M}$

Next, $\{p_\ell\}_{\ell=0}^3$ in (9) is calculated by using the result $\widehat{\mathbf{p}}$ in (10). Specifically,

- If $\ell = 0$, then
$$p_0 = \sum_{i=0}^{N-1} d_i = \widehat{p}_0 \tag{11}$$
- If $\ell = 1$ or 2, we have from Lemma 2 that
$$p_\ell = \sum_{i=0}^{N-1} d_i \cdot (w_i^\ell + u_m^\ell) = \widehat{p}_\ell + u_m^\ell \cdot \widehat{p}_0. \tag{12}$$
- If $\ell = 3$, then
$$\begin{aligned} p_3 &= \sum_{i=0}^{N-1} d_i \cdot (w_i + u_m)^3 \\ &= \widehat{p}_3 + u_m \cdot \widehat{p}_2 + u_m^2 \cdot \widehat{p}_1 + u_m^3 \cdot \widehat{p}_0 \\ &= \widehat{p}_3 + u_m \cdot (\widehat{p}_2 + u_m \cdot (\widehat{p}_1 + u_m \cdot \widehat{p}_0)). \end{aligned} \tag{13}$$

Fig. 2 shows the detailed process of calculating $\mathbf{p}$ from $\widehat{\mathbf{p}}$. The four black circles located at the bottom represent $\widehat{p}_0, \widehat{p}_1, \widehat{p}_2$, and $\widehat{p}_3$, respectively. According to (11), we have $p_0 = \widehat{p}_0$. The other three black circles represent p_1, p_2 and p_3, respectively. All white circles in Fig. 2 denote intermediate results that need to be calculated. In addition, each circle is obtained by summing over all the circles pointing to that circle, and the dashed arrows indicate that the circle is multiplied by a constant upon summation. It can be observed that when all elements in $\widehat{\mathbf{p}}$ are known, calculating all parity symbols requires a total of four additions and four multiplications over $\mathbb{F}_{2^{2M}}$. Here all multiplication factors are u_m. Lemma 4 gives the corresponding operation number over $\mathbb{F}_{2^M}$.

Lemma 4. *For any $a \in \mathbb{F}_{2^{2M}}$, the calculation of multiplying u_m by a only requires one addition and one multiplication over $\mathbb{F}_{2^M}$.*

Proof. Let $a = a_0 + a_1 \cdot u_m$ where a_0, a_1 are both in $\mathbb{F}_{2^M}$, we have from Lemma 1 that $u_m \cdot (a_0 + a_1 \cdot u_m) = (a_1 \cdot v_{M-1}) + (a_0 + a_1) \cdot u_m$, where $a_1 \cdot v_{M-1} \in \mathbb{F}_{2^M}, a_0 + a_1 \in \mathbb{F}_{2^M}$. It is clear that only $a_1 \cdot v_{M-1}$ and $a_0 + a_1$ need to be calculated. Since these operations are performed in $\mathbb{F}_{2^M}$, this completes the proof. □

In summary, the proposed encoding algorithm can be organized as follows:

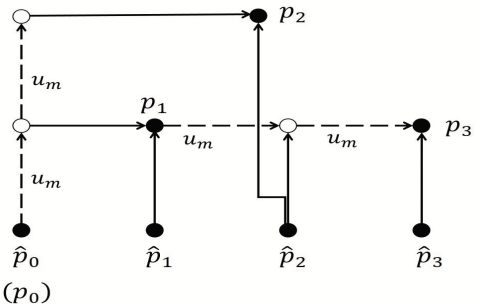

Fig. 2: Diagram of calculating $\mathbf{p}$ from $\widehat{\mathbf{p}}$.

Step E.1: Calculate $\widehat{\mathbf{p}}$ from (10) according to the RM-based algorithm proposed in [12].
Step E.2: Calculate $\mathbf{p}$ via Fig. 2.

The proposed encoding simultaneously deals with $2N$ symbols in $\mathbb{F}_{2^M}$.

4.2 Decoding

This paper only considers the case of all erasure. The case of non-all erasure can be derived from the case of all erasure. When all erased symbols are in the parity vector $\mathbf{p}$, the proposed encoding can perform the decoding directly. In the following, at least one symbol in the information vector $\mathbf{d}$ is erased by default. Note that the proposed decoding scheme also processes $2N$ symbols in $\mathbb{F}_{2^M}$ simultaneously, corresponding to the codeword (generated in the previous section) after deleting any four symbols.

Before describing the specific decoding process, we obtain the following formula by summarizing the calculation of (11)-(13):

$$\mathbf{0} = \widehat{H} \cdot (\mathbf{p}|\mathbf{d})^{\mathrm{T}}, \tag{14}$$

where

$$\widehat{H} = \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & \\ u_m & 1 & 0 & 0 & \\ u_m^2 & 0 & 1 & 0 & Vand_4(w_0, w_1, \cdots, w_{N-1}) \\ u_m^3 & u_m^2 & u_m & 1 & \end{array} \right). \tag{15}$$

Clearly, $\widehat{H}$ can be considered as a parity-check matrix equivalent to H in (5), since they produce the same codeword from $\mathbf{d}$. According to the decoding process described in Sec. 2.2, we then have two different parity-check matrix options for decoding.

When the parity-check matrix H in Sec. 2.2 is replaced by $\widehat{H}$ for decoding, Step C.1 can be completed efficiently through the RM-based algorithm in [12] and Lemma 4. At this time, Step C.2 only operates in $\mathbb{F}_{2^M}$ if all erased symbols are in $\mathbf{d}$. However, if there are erased symbols in $\mathbf{p}$, Step C.2 may generate a large number of operations in $\mathbb{F}_{2^{2M}}$. This can greatly reduce decoding efficiency. Although the computational efficiency in Step C.2 has less effect on the overall decoding as N increases, optimizing it when N is small is necessary. Next, we give a detailed decoding scheme for the case of erased symbols in $\mathbf{p}$. Note that, in this paper, any Vandermonde linear system is solved using lower–upper LU factorization [19], which results in fewer operations than the plain Gaussian elimination [13].

Except when all erased symbols are in $\mathbf{p}$ (the encoding can perform the decoding directly), there are 14 cases to be analyzed. By using Lemma 3, Lemma 4 and Lemma 5 to convert all operations to $\mathbb{F}_{2^M}$ for counting and comparison, we summarized which parity-check matrix is better for some cases. Thus, these 14 cases are split into two categories. We refer to the decoding method using $\widehat{H}$ and H as a parity-check matrix as "$\widehat{H}$-based" and "H-based", respectively. In each case, there may exist a Vandermonde sub-linear system which can be solved by using LU factorization [19]. This leads to a process that varies from case to case, and it requires a combination of LU factorization and Gaussian elimination. For simplicity, we classify those cases with similar processes as one family. The detailed decoding scheme is given below.

Lemma 5. *Multiplying any $a \in \mathbb{F}_{2^{2M}}$ by a fixed multiplication factor $b \in \mathbb{F}_{2^{2M}}$ requires three addition and three multiplication over $\mathbb{F}_{2^M}$.*

Proof. Let $a = a_0 + a_1 \cdot u_m$ and $b = b_0 + b_1 \cdot u_m$ where all a_0, a_1, b_0, b_1 are in $\mathbb{F}_{2^M}$. Then we have from Lemma 1 that $a \cdot b = (a_0 + a_1 \cdot u_m) \cdot (b_0 + b_1 \cdot u_m) = a_0 b_0 + a_1 b_1 v_{M-1} + (a_0 b_1 + a_1 b_0 + a_1 b_1) \cdot u_m = a_0 b_0 + a_1 b_1 v_{M-1} + ((a_0 + a_1)(b_0 + b_1) - a_0 b_0) \cdot u_m$. Since b and v_{M-1} are fixed, only three multiplications and three additions over $\mathbb{F}_{2^M}$ are required. □

4.2.1 $\widehat{H}$-based

In this method, Step C.1 can be accelerated by using Lemma 4, and the RM-based algorithm in [12], then Step C.2 is performed as follows: (in brackets are all erased symbols in $\mathbf{p}$.)

1) Case (p_1, p_2, p_3): All erased symbols are solved by Gaussian elimination. In this case,
$$H_e = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & w_i \\ 0 & 1 & 0 & w_i \\ u_m^2 & u_m & 1 & w_i \end{pmatrix}, \text{ where } 0 \le i < N. \quad (16)$$
One can obtain the last erased symbol based on the first row of H_e and then use Gaussian elimination to calculate the first, second, and third erased symbols in order from the second row to the fourth row. Note that the operation of multiplying w_i needs to be performed only once.
2) Case $(p_0), (p_3), (p_0, p_3), (p_1, p_3), (p_2, p_3), (p_0, p_1, p_3), (p_0, p_2, p_3)$: Many erased symbols are first obtained by solving a Vandermonde sub-linear subsystem, and then all remaining erased symbols are obtained by Gaussian elimination. For example, if p_0 and p_3 are erased, then
$$H_e = \begin{pmatrix} 1 & 0 & 1 & 1 \\ u_m & 0 & w_i & w_j \\ u_m^2 & 0 & w_i^2 & w_j^2 \\ u_m^3 & 1 & w_i^3 & w_j^3 \end{pmatrix}, \text{ where } 0 \le i < j < N. \quad (17)$$
One can obtain the first, third, and fourth erased symbols based on the first three rows of H_e (forming a Vandermonde sub-linear system), and then calculate the second erased symbol by Gaussian elimination based on the last row of H_e.
3) Case $(p_1), (p_2)$: The solutions in these two cases are similar. Taking the former as an example, we have that $H_e = \begin{pmatrix} 0 & \\ 1 & V \in \mathbb{F}_{2^M}^{3\times 3} \\ 0 & \\ u_m^2 & \mathbf{a} \in \mathbb{F}_{2^M}^3 \end{pmatrix}$, where V is a Vandermonde matrix. Further, H_e can be decomposed into
$$H_e = \begin{pmatrix} & & & 0 \\ & V & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{c} \in \mathbb{F}_{2^M}^{3\times 1} & I_{3\times 3} \\ u_m^2 & \mathbf{a} \end{pmatrix}, \quad (18)$$
where $\mathbf{c} = V^{-1} \cdot (0, 1, 0)^{\mathrm{T}}$. The above formula shows that Step C.2 can be solved in two steps. The first step is still to solve a Vandermonde linear system determined by V in (18). The second step can be completed by using Gaussian elimination on $\begin{pmatrix} \mathbf{c} \in \mathbb{F}_{2^M}^{3\times 1} & I_{3\times 3} \\ u_m^2 & \mathbf{a} \end{pmatrix}$ due to the existence of $I_{3\times 3}$. Specifically, in the second step, the first erased symbol is first obtained based on the last row, and then other erased symbols are obtained based on the first three rows.

4.2.2 H-based

This method is used for all the remaining cases, i.e. $(p_1, p_2), (p_0, p_2), (p_0, p_1)$ and (p_0, p_1, p_2). Step C.1 can be performed quickly by calling the proposed encoding. For Step C.2, some erased symbols are obtained by solving a linear subsystem, and then all remaining erased symbols are obtained by Gaussian elimination. For example, in the case of (p_1, p_2), we have
$$H_e = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & h_i & h_j \\ 0 & 1 & h_i^2 & h_j^2 \\ 0 & 0 & h_i^3 & h_j^3 \end{pmatrix}, \text{ where } 0 \le i < j < N. \quad (19)$$
One can obtain the last two erased symbols based on the first and fourth rows of H_e (forming a Vandermonde sub-linear system), and then calculate the first two erased symbols by Gaussian elimination based on the second and third rows of H_e.

5 Analysis and Comparison

In this section, we first analyze the computational complexity of the proposed encoding/decoding, and then compare them with other alternative algorithms.

For the proposed encoding, one can see from [12] that Step E.1 requires $3N + n - 6$ additions and $\frac{n^2+5n}{2} - 3$ multiplications over $\mathbb{F}_{2^{2M}}$. Note that $N = 2^n$. According to Lemma 3 and the fact that one addition over $\mathbb{F}_{2^{2M}}$ is equivalent to two additions over $\mathbb{F}_{2^M}$. Thus, Step E.1 requires $6N + 2n - 12$ additions and $n^2 + 5n - 6$ multiplications over $\mathbb{F}_{2^M}$. In addition, Step E.2 requires four additions and four multiplications over $\mathbb{F}_{2^{2M}}$. From Lemma 4, this is equivalent to 12 additions and four multiplications over $\mathbb{F}_{2^M}$. In summary, the average numbers of additions and multiplications over $\mathbb{F}_{2^M}$ are respectively $\frac{(6N+2n-12)+12}{2N} = 3 + \frac{n}{N}$ and $\frac{(n^2+5n-6)+4}{2N} = \frac{n^2+5n-2}{2N}$. Note that the information vector $\mathbf{d}$ has $2N$ symbols in $\mathbb{F}_{2^M}$, as shown in Sec. 4.1.

TABLE 1: Average numbers of additions/multiplications over $\mathbb{F}_{2^8}$ (all data are rounded)

N	8	16	32	64	128
Encoding					
[12]	4.38/2.38	3.75/1.56	3.41/1.00	3.22/0.63	3.12/0.38
[14]	3.00/1.50	3.06/1.13	3.44/1.03	\	\
Our	3.38/1.38	3.25/1.06	3.16/0.75	3.09/0.5	3.05/0.32
Normalized with respect to [12]					
[14]	0.68/0.63	0.82/0.72	1.01/1.03	\	\
Our	0.77/0.58	0.87/0.68	0.93/0.75	0.96/0.80	0.98/0.84
Average-case decoding					
[12]	5.44/3.21	4.28/1.98	3.68/1.21	3.36/0.73	3.19/0.44
[14]	4.33/2.63	3.86/1.79	3.89/1.39	\	\
Our	5.13/2.66	4.20/1.79	3.65/1.13	3.35/0.69	3.18/0.42
Normalized with respect to [12]					
[14]	0.80/0.82	0.90/0.90	1.05/1.15	\	\
Our	0.94/0.83	0.98/0.90	0.99/0.93	1.00/0.95	1.00/0.95

TABLE 1 lists the average number of operations for different encoding/decoding algorithms, in which the finite fields used are all set to be $\mathbb{F}_{2^8}$. To facilitate comparison, the numbers of operations in the last two algorithms are normalized to that in [12]. It can be observed that the number of operations in the proposed encoding is significantly less than that in [12], especially when N is small. This is due to the fact that the size of the RM transform in our scheme is smaller than that in [12], leading to fewer operations. Moreover, Step E.2 requires fewer operations than Step C.2. [14] has an encoding complexity close to ours. However, it requires significantly more operations than ours at $N = 32$. This is because the RM transform of size more than 32 is needed in [14] to match a 32-column Vandermonde matrix without singular sub-matrices instead of the RM transform with size exactly 32 as in the proposed scheme. Furthermore, the maximum feasible N in [14] is 52, and the corresponding H is not explicitly given. In contrast, the proposed construction can make N up to $2^M = 256$ in this case, and H has an explicit construction.

For decoding, erasures are typically evenly distributed among all symbols. TABLE 2 lists the number of operations over $\mathbb{F}_{2^M}$ required by the proposed decoding in different cases. Specifically, we split the decoding procedure into two components: the first performs the RM-based algorithm with size N, and the second performs all the remaining operations. The first component produces the same number of operations as Step E.1 in the proposed encoding, as shown in the upper part of TABLE 2. In addition, the second component produces the number of operations independent of N, as shown in the lower part of TABLE 2. In particular, the lower part of TABLE 2 shows that the number of operations in different cases can vary by almost twice, which is caused by H_e containing different columns of the identity matrix. Using the result in TABLE 2, we then give the average-case complexity of the proposed decoding, shown in the lower part of TABLE 1. It can be seen that the proposed decoding also has less number of operations than the algorithm in [12]. This is mainly caused by the RM transform with a smaller size. The decoding comparison between the proposed scheme and [14] is similar to that of the encoding comparison. Again, the algorithm in [14] requires N to be no more than 52 and does not give the explicit parity-check matrix H.

TABLE 2: Number of additions/multiplications over $\mathbb{F}_{2^M}$ required by the proposed decoding in different cases

Component 1: # of operations generated by RM-based algorithm $(6N + 2n - 12)/(n^2 + 5n - 6)$				
Component 2: # of other operations in different cases				
$()$	(p_0)	(p_1)	(p_2)	(p_3)
45/29	45/29	42/30	43/31	39/23
(p_2, p_3)	(p_1, p_3)	(p_0, p_3)	(p_1, p_2)	(p_0, p_2)
29/17	29/17	37/23	46/22	43/21
(p_0, p_1)	(p_1, p_2, p_3)	(p_0, p_2, p_3)	(p_0, p_1, p_3)	(p_0, p_1, p_2)
43/21	24/12	27/15	27/15	33/13

From TABLE 1, one can see that the advantages of addition and multiplication complexities in the proposed scheme do not always occur simultaneously. For a comprehensive comparison, we next follow the efficiency ratio η of multiplication and addition realized by Single Instruction Multiple Data (SIMD) technology, i.e., one multiplication over $\mathbb{F}_{2^8}$ is equivalent to η additions over $\mathbb{F}_{2^8}$. Based on our experience in practice [12], η can be set to four. Then we define the total computational complexity as the sum of the average number of additions and η times the average number of multiplications. Fig. 3 shows the comparisons of different RM-based RS encoding over $\mathbb{F}_{2^8}$, where η has more possible values. It can be observed that the proposed encoding always has the best total computational complexity, regardless of the value of η. The ratio η only affects the total computational complexity of each algorithm, not the differences between algorithms. Note that, in Fig. 3, we have added the general RS encoding proposed in [13], which also improved the original RM transform-based RS encoding. The difference between [14] and this paper is that [14] uses a conventional Vandermonde matrix as the parity-check matrix, so there is no identity matrix in its parity-check matrix. This results in the size of the RM transform in [13] being larger than ours. In the range of parameters considered in Fig. 3, the computational complexity of the proposed encoding is on average 25.8%, 13.6%, 6.9%, lower than those in [12], [13], and [14], respectively.

Fig. 4 shows the comparisons of different RM-based RS decoding over $\mathbb{F}_{2^8}$. It provides the improvement of different decoding algorithms over the algorithm in [12] in the average and worst cases. The average-case decoding in [13] always has the same computational complexity as its worst-case decoding since it has no identity matrix in the parity-check matrix. Hence, we only compare it with the worst-case decoding in [12]. It can be observed from Fig. 4 that when $N \geq 32$, the improvement of all algorithms is not significant. When $N < 32$, the decoding in [14] has the best improvement. However, its decoding performance is much worse at $N = 32$. This is because the RM transform of size more than 32 is needed in [14] to match a searched 32-column Vandermonde matrix instead of the RM transform with a size of exactly 32 as in the proposed scheme. Importantly, the parity-check matrix in [14] is not explicit and the feasible value of N cannot exceed 52, which was found by computer search considering all possible (1050 different) constructions of $\mathbb{F}_{2^8}$ [14]. Thus, the proposed scheme in this paper is a good alternative to the one in [14]. Compared

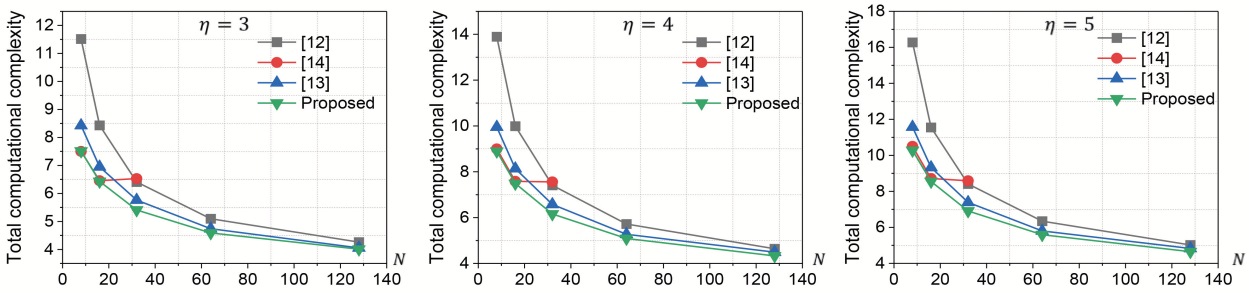

Fig. 3: Comparisons of different RM-based RS encoding over $\mathbb{F}_{2^8}$ (the total computational complexity defined as the sum of the average number of additions and η times the average number of multiplications).

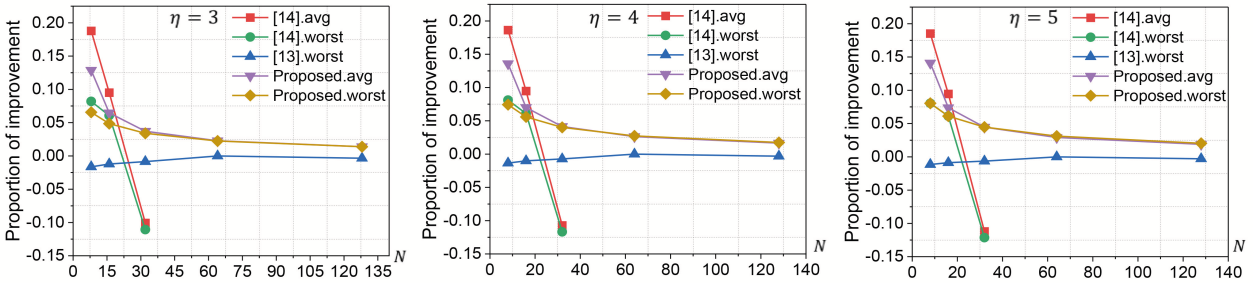

Fig. 4: Comparisons of different RM-based RS decoding over $\mathbb{F}_{2^8}$ (the vertical axis represents the proportion of improvement in total computational complexity for each scheme compared to [12]).

with [13], the decoding scheme proposed in this paper is always superior.

It is worth mentioning that the maximum feasible codeword length of the proposed scheme is $N + 4 = 2^M + 4$ when performing efficiently on $\mathbb{F}_{2^M}$. This is the codeword length of four symbols more than the extended RS code over $\mathbb{F}_{2^M}$ [15]. When $N = 2^M$, the algorithms in [12], [13] must be performed on a larger finite field than $\mathbb{F}_{2^M}$. Smaller fields with simpler arithmetic implementations lead to the obvious superiority of the proposed scheme.

6 Conclusion

In this paper, we first construct a systematic Vandermonde matrix from the perspective of algebraic construction, and then propose a new RM-based RS encoding/erasure decoding algorithm. To highlight the advantages of the proposed construction, we compare it with other RM-based RS codes in terms of encoding/erasure decoding. The results show the proposed construction is a good alternative to other RM-based codes because of the more explicit construction, a wider range of codeword lengths, and the competitive encoding/erasure decoding performance.

Acknowledgments

The authors would like to thank the associate editor, and the anonymous reviewers for their valuable comments and suggestions that helped us in improving this paper. In addition, the authors would like to acknowledge the support of the National Key Research and Development Program of China under Grant 2022YFA1004902.

References

[1] S.-J. Lin, A. Alloum, and T. Y. Al-Naffouri, "Raid-6 Reed-Solomon codes with asymptotically optimal arithmetic complexities," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2016, pp. 1–5.

[2] S.-J. Lin, "An encoding algorithm of triply extended Reed–Solomon codes with asymptotically optimal complexities," *IEEE Transactions on Communications*, vol. 66, no. 8, pp. 3235–3244, 2017.

[3] S. Muralidhar, W. Lloyd, S. Roy, C. Hill, E. Lin, W. Liu, S. Pan, S. Shankar, V. Sivakumar, L. Tang *et al.*, "f4: Facebook's warm BLOB storage system," in *Proc. of USENIX OSDI*, 2014, pp. 383–398.

[4] G. Kumar, N. Dukkipati, K. Jang, H. M. Wassel, X. Wu, B. Montazeri, Y. Wang, K. Springborn, C. Alfeld, M. Ryan *et al.*, "Swift: Delay is simple and effective for congestion control in the datacenter," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 514–528.

[5] Y. Arafa, A. Barai, M. Zheng, and A.-H. A. Badawy, "Fault tolerance performance evaluation of large-scale distributed storage systems HDFS and Ceph case study," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*. IEEE, 2018, pp. 1–7.

[6] L. Jin, "Explicit construction of optimal locally recoverable codes of distance 5 and 6 via binary constant weight codes," *IEEE Transactions on Information Theory*, vol. 65, no. 8, pp. 4658–4663, 2019.

[7] N. Prakash, V. Abdrashitov, and M. Médard, "The storage versus repair-bandwidth trade-off for clustered storage systems," *IEEE Transactions on Information Theory*, vol. 64, no. 8, pp. 5783–5805, 2018.

[8] L. Holzbaur, S. Puchinger, E. Yaakobi, and A. Wachter-Zeh, "Partial MDS codes with regeneration," *IEEE Transactions on Information Theory*, vol. 67, no. 10, pp. 6425–6441, 2021.

[9] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[10] Q. Yu and A. S. Avestimehr, "Coded computing for resilient, secure, and privacy-preserving distributed matrix multiplication," *IEEE Transactions on Communications*, vol. 69, no. 1, pp. 59–72, 2020.

[11] M. Blaum and R. M. Roth, "On lowest density MDS codes," *IEEE Transactions on Information Theory*, vol. 45, no. 1, pp. 46–59, 1999.
[12] L. Yu, Z. Lin, S.-J. Lin, Y. S. Han, and N. Yu, "Fast encoding algorithms for Reed–Solomon codes with between four and seven parity symbols," *IEEE Transactions on Computers*, vol. 69, no. 5, pp. 699–705, 2020.
[13] L. Yu, S.-J. Lin, H. Hou, and Z. Li, "Reed-Solomon coding algorithms based on Reed-Muller transform for any number of parities," *IEEE Transactions on Computers*, vol. 72, no. 9, pp. 2677–2688, 2023.
[14] Y. J. Tang and X. Zhang, "Fast en/decoding of Reed-Solomon codes for failure recovery," *IEEE Transactions on Computers*, vol. 71, no. 3, pp. 724–735, 2021.
[15] S. B. Wicker and V. K. Bhargava, *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.
[16] D. G. Cantor, "On arithmetical algorithms over finite fields," *Journal of Combinatorial Theory, Series A*, vol. 50, no. 2, pp. 285–300, 1989.
[17] G. L. Mullen and D. Panario, *Handbook of finite fields*. CRC press Boca Raton, 2013, vol. 17.
[18] N. Kolokotronis, K. Limniotis, and N. Kalouptsidis, "Lower bounds on sequence complexity via generalised Vandermonde determinants," in *International Conference on Sequences and Their Applications*. Springer, 2006, pp. 271–284.
[19] S.-L. Yang, "On the LU factorization of the Vandermonde matrix," *Discrete applied mathematics*, vol. 146, no. 1, pp. 102–105, 2005.

Leilei Yu received the B.Eng. degree in electronic information engineering from the Tianjin University of Technology, Tianjin, China, in 2015, and the Ph.D. degree in cyberspace security from the University of Science and Technology of China, Hefei, China, in 2021. From 2021 to 2022, he was a cybersecurity researcher with the Purple Mountain Laboratories, Nanjing, China. He is currently a postdoctoral research fellow with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China. His research focuses on coding theory and high-performance computing.

Yunghsiang S. Han (S'90-M'93-SM'08-F'11) was born in Taipei, Taiwan, 1962. He received B.Sc. and M.Sc. degrees in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respectively, and a Ph.D. degree from the School of Computer and Information Science, Syracuse University, Syracuse, NY, in 1993. He was from 1986 to 1988 a lecturer at Ming-Hsin Engineering College, Hsinchu, Taiwan. He was a teaching assistant from 1989 to 1992, and a research associate in the School of Computer and Information Science, Syracuse University from 1992 to 1993. He was, from 1993 to 1997, an Associate Professor in the Department of Electronic Engineering at Hua Fan College of Humanities and Technology, Taipei Hsien, Taiwan. He was with the Department of Computer Science and Information Engineering at National Chi Nan University, Nantou, Taiwan from 1997 to 2004. He was promoted to Professor in 1998. He was a visiting scholar in the Department of Electrical Engineering at University of Hawaii at Manoa, HI from June to October 2001, the SUPRIA visiting research scholar in the Department of Electrical Engineering and Computer Science and CASE center at Syracuse University, NY from September 2002 to January 2004 and July 2012 to June 2013, and the visiting scholar in the Department of Electrical and Computer Engineering at University of Texas at Austin, TX from August 2008 to June 2009. He was with the Graduate Institute of Communication Engineering at National Taipei University, Taipei, Taiwan from August 2004 to July 2010. From August 2010 to January 2017, he was with the Department of Electrical Engineering at National Taiwan University of Science and Technology as Chair Professor. From February 2017 to February 2021, he was with School of Electrical Engineering & Intelligentization at Dongguan University of Technology, China. Now, he is with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China and as a consultant of Huawei Technology company. His research interests are in error-control coding, wireless networks, and security.

Dr. Han was a winner of the 1994 Syracuse University Doctoral Prize and a Fellow of IEEE. One of his papers won the prestigious 2013 ACM CCS Test-of-Time Award in cybersecurity.