Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 29 (2026), No. 10-C, pp. 4671–4682
DOI : 10.47974/JDMSC-3082

Construction of error-correcting codes from finite geometries and their applications in secure data storage systems

Simranjeet Nanda [†]
Centre for Research Impact & Outcome
Chitkara University Institute of Engineering and Technology
Chitkara University
Rajpura 140401
Punjab
India

Arsh Goyal *
Quantum University Research Center
Quantum University
Roorkee 247667
Uttarakhand
India

Karan Pandre [§]
Centre for Multidisciplinary Research
Anurag University
Hyderabad 500088
Telangana
India

Arpita A. Prajapati [‡]
Faculty of Engineering
Gokul Global University
Sidhpur 384151
Gujarat
India

[†] *ORCID-ID:* 0009-0005-6893-8585
[*] *ORCID-ID:* 0009-0009-1847-4064
[§] *ORCID-ID:* 0009-0001-0631-4006
[‡] *ORCID-ID:* 0009-0003-8878-9762

[†] *E-mail:* simranjeet.nanda.orp@chitkara.edu.in
[*] *E-mail:* aarshgoyal@gmail.com (Corresponding Author)
[§] *E-mail:* KaranPandre2024@protonmail.com
[‡] *E-mail:* aaprajapati.ce.hcet@gokuluniversity.ac.in

© Taru Publications

Tirukoti Sudha Rani [@]
Department of Computer Science & Engineering
Aditya University
Surampalem
Kakinada District 533437
Andhra Pradesh
India

Hitendra Garg [#]
Department of Computer Engineering & Applications
GLA University
Mathura 281406
Uttar Pradesh
India

Abstract

The fast expanding distributed and cloud-based storage systems have escalated the necessity to have strong error-correcting protocols that guarantee data integrity, reliability, and security. The conventional coding schemes are usually limited in scalability and structural efficiency when they are implemented in massive storage settings. The paper presents a new model to build error-correcting codes based on finite geometries, with the help of the combinatorial constructions like projective and affine planes, to become more redundancy and fault-tolerant. The algorithm combines geometry-based code construction with algebraic encoding and decoding, which allows to effectively detect and correct errors in high-noise scenarios. Experimental assessments reveal that the model proposed has an error correction rate of 96.7% which is more reliable than classical Reed-Solomon codes and the error correction rate less than the classical error correction rate. The framework also provides a high level of resiliency to data corruption and provides a scalable storage architecture. These results suggest that finite geometry-based codes have a promising future in building secure, efficient and high-performance data storage systems.

Subject Classification: *68M25, 68M12.*

Keywords: *Finite geometry codes, Error-correcting codes, Projective planes, Affine geometry, Secure data storage, Reed–Solomon codes.*

1. Introduction

The rapid growth of creating digital data has imposed unprecedented loads on the present-day storage infrastructures, especially cloud-based and distributed systems. Error-correcting codes (ECCs) are needed to ensure data integrity in such systems, and can effectively detect and mitigate bit-level errors due to hardware faults, transmission noise, and adversarial attacks [1]. Traditional coding schemes like Hamming codes and

[@] *E-mail:* **sudharani.tirukoti@adityauniversity.in**
[#] *E-mail:* **hitendra.garg@gla.ac.in**

cyclic codes have traditionally dealt with such issues but have inherent limitations when used in dense and multi node storage systems [2]. Although used in optical storage and RAID systems, Reed-Solomon (RS) codes, though based on the algebraic polynomials, have fixed trade-offs between code rate and error correction capability, and thus cannot be easily changed to dynamic storage requirements [3]. Likewise, Low-Density Parity-Check (LDPC) codes are also able to achieve close to Shannon-limit performance, but must employ intensive iterative decoding algorithms that augment the computational load of large-scale architectures [4]. This increasing inefficiency of conventional codes has impelled researchers to seek alternative mathematical basis in the construction of ECC. The code construction based on finite geometry, namely, projective and affine geometric structures, provides a mathematically rich and combinatorically systematic code construction, making it possible to derive codes with good structural properties, such as high minimum distance, uniform weight distributions, and efficient parity-check matrix constructions [5]. Codes based on finite geometries enjoy natural regularity and distance properties, which is why they are an appropriate solution to applications with a high fault tolerance requirement as well as a high deterministic redundancy [6]. In addition, they are highly structured and thus can be implemented with efficient hardware to support the new non-volatile memory and distributed object storage paradigms [7]. The main aims of this work are to: (i) produce a logical framework of building the error-correcting codes on projective and affine finite geometries; (ii) examine the parameters of algebra like the minimum distance, code rate, and redundancy; (iii) implement such codes in a data store infrastructure maintaining security; and (iv) test the proposed structure against the current state-of-the-art baseline codes by providing rigorous simulation experiment.

2. System Architecture and Framework Design

The proposed system architecture expands finite geometry-based error-correcting codes into a multi-layered reliable storage system, comprising of geometric code construction, algebraic encoding pipelines and cryptographic integrity verification units to provide high-reliability distributed data storage with tolerance to faults and noise under realistic noise and failure scenarios [8].

2.1 *Overall Architecture of Secure Storage System*

The secure storage architecture has three main layers such as data ingestion layer, encoding and redundancy layer and the storage distribution layer. Raw data blocks are passed to the ingestion layer where they are broken up and made ready to be encoded. The encoding layer encodes codewords using parity matrices derived with finite geometry and these are further spread over storage nodes. A monitoring layer constantly checks the

integrity of the block by using syndrome-based checks and initiates error recovery in case deviations are detected [9].

2.2 *Data Encoding and Decoding Workflow*

Encoding Data encoding starts with dividing the input bitstreams into message blocks of fixed length of dimension k. A generator matrix, based on finite geometric incidence matrices, multiplies each block to generate a codeword of length n, and decoding uses an algorithm based on syndromes: received vectors are multiplied by the parity-check matrix H to obtain syndromes, non-zero syndromes are used to invoke error localization and correction algorithms. The complete workflow is optimized on the basis of the geometric structure properties through the use of lookup tables that reduce the amount of computation [10].

2.3 *Integration of Finite Geometry Structures*

The parity-check matrix H is built using the incidence matrix of a finite projective plane PG(2, q), the rows of which are the lines, and whose columns are the points. The result is an LDPC-like structured sparsity and minimum distance. The equations that are used to describe the geometric integration are the following.

$$H = Inc\big(PG(2,q)\big), H \in \{0,1\}^{q^2+q+1} \times (q^2+q+1) \quad (1)$$

In (1) H is the incidence matrix of the projective plane PG(2,q), and q is a prime power, and H is a square binary matrix of order q 2 + q + 1.

$$rank(H) = q^2 + q + 1 - k, d_{\min} \geq q + 2 \quad (2)$$

The rank of H and minimum Hamming distance d_{min} is given in Eq. (2), and is used to assure an error correction rate of at least errors of the code.

$$C_{PG} = \{x \in GF(q)^n : H \cdot x^{\mathrm{T}} = 0\}, n = q^2 + q + 1 \quad (3)$$

The null space of H over GF(q) is called CPG and defined by the codeword set Cpg.

$$G \cdot H^{\mathrm{T}} = 0 \implies \dim(C_{PG}) = n - rank(H) = k \quad (4)$$

The orthogonality of G and H is verified by Eq. (4) and thus the code dimension, k, is n - rank of H, which is a common property that can be stated about linear coding theory.

2.4 *Security and Reliability Mechanisms*

Multi-layer redundancy and the use of hash-based integrity checks at every node of storage provide security. The system identifies tampering of data through comparison of the stored codeword syndromes with the expected values. Adaptive erasure correction based on the geometric structure of the dual code is used to improve reliability, with burst erasures or random bit errors of the code being corrected.

$$P_{fail} = \Pi_{\{i=0\}}^{\{t\}} [1 - (n\ choose\ i) \cdot p^i \cdot (1-p)^{n-i}] \quad (5)$$

The binomial distribution of error patterns in decoding is used to determine the probability of decoding failure P_{fail} in a t-error-correcting code, as shown in Eq. (5), based on the binomial distribution of the error patterns.

$$R_{secure} = (1 - \delta) \cdot \left(\frac{k}{n}\right), \delta = \frac{H(p)}{log^2(q)} \quad (6)$$

The code of secure rate R secure is expressed in the form of Eq. (6) that considers security margin 1, H(p) binary entropy, and q the field order at which the code is constructed.

$$MTTR = (\lambda \cdot d_{min})^{-1} \quad (7)$$

In which, λ = node failure rate.

The geometric code properties of a system are connected with system reliability through the model of mean time to recovery shown in the following equation (7): MTTR = -1/failure rate 1/dmin.

$$Integrity_{Score} = \left(\frac{1}{N}\right) \Sigma_{\{i=1\}}^{\{N\}} [S_i = 0\ ?\ 1 :\ 0], S_i = H \cdot r_i{}^{r} \quad (8)$$

In Eq. (8), the system integrity score is calculated as the proportion of storage blocks which have zero syndrome S_i, r_i the received vector of node i and N the total number of nodes.

3. Finite Geometry-Based Code Construction

Finite geometry offers a strong combinatorial theory of how to build structured error-correcting codes with algebraic properties that are rigorously known to be strong. Using the incidence relations on projective and affine spaces over finite fields, codes are produced that have optimum minimum distance properties, an efficient encoding complexity, and natural regularity in their parity-check matrices, all beneficial in high-performance deployment of storage systems.

3.1 *Projective Geometry-Based Code Design*

The codes of projective geometry (PG-codes) are based on the incidence structure of the projective plane PG(2, q). The rows of the parity matrix are each projective lines of q+1 points. The resulting code has the highest possible minimum distance per length and dimension and as such it is extremely efficient in burst and random error correction in storage mediums.

$$PG(m,q) has \frac{q^{m+1} - 1}{q - 1} points$$

$$\frac{q^{m+1} - 1}{q^2 - 1} \cdot \frac{q^2 - 1}{q - 1} lines$$

This determines the combinatorics of PG(m,q), determining exactly the number of points and lines with which the rows and columns of the parity-check matrix are built.

$$\mathrm{H_{PG}} = \left[\mathrm{h_{\{ij\}}}\right] \tag{9}$$

$h_{\{ij\}} = 1\ iff$ point j lies on line i in PG(2, q)

HPG is the binary incidence matrix of PG(2, q), with a one entry encoding geometric incidence, a direct encoding of parity constraints of projective geometry.

$d_min(C_PG) = q + 2$, for q = prime power,

$$C_{PG} \subseteq GF(q)^{q^2+q+1} \tag{10}$$

The Hamming distance of the PG-code is at least q + 2, ensuring that it allows one to correct up to ⌊(q+1)/2) errors, which has been shown to be a tight bound in finite geometry combinatorics.

$$Rate(C_{PG}) = \frac{k_{PG}}{n_{PG}} = \frac{q^2+q+1-rank(H)}{q^2+q+1} \tag{11}$$

The code rate measures the ratio of information bits to length of the codeword and trades off redundancy and efficiency based on the rank deficiency of the geometric incidence matrix.

3.2 *Affine Geometry-Based Code Construction*

The affine plane AG (2, q) is used to form affine geometry codes (AG-codes), forming a denser and more uniform incidence structure by deleting the line at infinity in PG(2, q). The parity-check matrices that are generated are quasi-cyclic making it easy to implement the hardware encoding. AG-codes attain a little better code rates than PG-codes and have good minimum distance properties that can be used in distributed storage.

AG(2, q) contains q 2 points, q 2(q+1)/... parallel classes, each with q parallel lines.

The affine plane AG(2, q) divides its q^{2} points into q+1 parallel classes each with q non-intersecting lines to give the quasi-cyclic structure of the code matrix.

$$H_{AG} \in \{0,1\}^{q(q+1)} \times q^2 \tag{12}$$

The weight of the row = q, of the col = q+1.

The HAG is a uniformly weighted (rectangular) binary matrix with uniform row and column weights, and therefore has a regular LDPC structure, which is easy to code and decode using belief-propagation decoding algorithms.

$$d_{\min(C_{AG})} \geq q + 1 \tag{13}$$

compared to $d_{\min(C_{PG})} = q + 2$

AG-codes have the minimum distance of at least q+1, a little less than that of PG-codes, but have a higher code rate because of the lack of the projective line at infinity.

$$C_{AG} = Null(H_{AG}) over\ GF(q) \quad (14)$$

$$k_{AG} = q^2 - rank(H_{AG}) \quad (15)$$

The null space of H AG over GF(q) is denoted AG-code and its dimension k AG is calculated as the difference between the dimension of H AG and the total number of the coordinates of the codewords.

3.3 *Code Parameters (Minimum Distance, Rate, Redundancy)*

The geometric structure of the space in which finite geometry codes are being used allows the derivation of the fundamental parameters of finite geometry codes in an analytical fashion. The Singleton and Plotkin bounds relate Minimum distance, code rate and redundancy. These parameters are accurately calculable in both the case of PG and AG codes, and allow the field order q to be selected for a particular set of storage reliability and efficiency needs.

Singleton Bound: $d_{min} \leq n - k + 1$, obtained with MDS codes.

The Singleton bound gives an upper bound on dmin of a given (n,k) code; MDS codes are attaining this bound, and FG-codes are asymptotically equal to the bound as q increases.

Redundancy $r = n - k = rank(H), r_{PG} = rank(H_{PG}) \leq q^2 + q + 1$ (16)

Redundancy r is the amount of parity bits and this is defined by the rank of H. In case of PG-codes, r is no more than q 2 +q +1 giving good error coverage at a limited overhead.

Rate-Distance Trade-off: $R \cdot d_{\min} \leq n \cdot (1 - R)$ (17)

This inequality embodies the basic rate-distance-trade-off: the larger the code rate, the less redundancy and minimum distance, the less error correction power.

$$Correction\ Capability\ t = \left\lfloor \frac{d_{\min}-1}{2} \right\rfloor, erasure\ capability = d_{\min}-1 \quad (18)$$

A code of minimum distance dmin is able to detect up to dmin-1 erasures, and to correct up to t random errors, which are important parameters to assess the storage fault tolerance capacity.

4. Methodology and Experimental Setup

The experimental analysis was done on a simulated distributed storage system that had 32 virtualized storage nodes, each having a storage capacity of 10 GB, and connected through a 10 Gbps network fabric. The dataset was synthetic binary data blocks of 100MB to 10GB, which were created with the pseudo-random bit sequence (PRBS) generator to create realistic workloads with variable entropy. Burst erasure models and

additive white Gaussian noise (AWGN) were used to model bit corruption, node failures with different bit error rates (BER) ranging between 0.01 to 0.20. The code proposed via the finite geometry (FG-Code) was written in Python 3.10 with the NumPy and SciPy libraries to operate matrices over GF(q), where q is 4, 8, 16 and 32, and tested. Such simulation tools as the AFF3CT open-source channel coding library of ReedSolomon and LDPC baseline comparisons were involved. All the experiments were run on a server with Intel Xeon Gold 6248R (24 cores, 3.0 GHz) and 128 GB of ECC RAM and an NVIDIA A100 acceleration processor to do parallel decoding. The performance was measured based on five main measures including error correction accuracy (ECA), error correction latency (DL), throughput (TP), fault tolerance ratio (FTR), and storage overhead ratio (SOR). Comparisons with baseline codes (ReedSolomon 255, 223) and Turbo codes (two recursive systematic convolutional encoders) and standard Hamming (63, 57) codes were made. All experiments were done 50 times and average measures with 95 percent confidence are presented in order to ensure statistical strength.

5. Results, Comparative Analysis, and Discussion

The experimental data fully confirm the effectiveness of the suggested framework of finite geometry-based error-correcting codes as compared to classical benchmarks in all five assessment aspects. The FG-Code has always outperformed the other two algorithms in error correction capabilities, minimized decoding latency, achieved best scalability, and fault tolerance, which validates that the algorithm is practically viable in deployment in secure high-performance distributed storage systems at scale.

5.1 *Quantitative Performance Evaluation*

Table 1 shows the general scores of the quantitative performance of all the five evaluation metrics of each coding method. Proposed FG-Code performs the best with a composite performance score of 96.7, Turbo codes with a performance score of 91.2, LDPC codes with 88.3, Reed Solomon with 84.6 and Hamming with 79.4. These findings affirm that the geometric structural properties of FG-Code are literally translated into high-performance coding under different testing conditions. The strength of the FG-Code lies in the most inaccuracy error correction and fault tolerance with the regularity of the projective geometry parity matrix giving structural strength. The gap between the proposed FG-Code and all baselines in terms of the performance is visually emphasized in Figure 1, which once again proves the superiority of the overall score of the proposed FG-Code.

Table 1

Quantitative Performance Comparison of Error-Correcting Code Methods

Method	ECA (%)	DL (ms)	TP (Gbps)	FTR (%)	SOR (%)
FG-Code (Proposed)	96.7	18.6	9.12	97.2	92.3
Reed–Solomon	84.6	22.9	7.43	84.6	85.7
LDPC	88.3	24.1	7.89	88.3	83.1
Turbo	91.2	31.5	6.21	90.1	80.4
Hamming	79.4	13.2	8.90	79.4	91.2

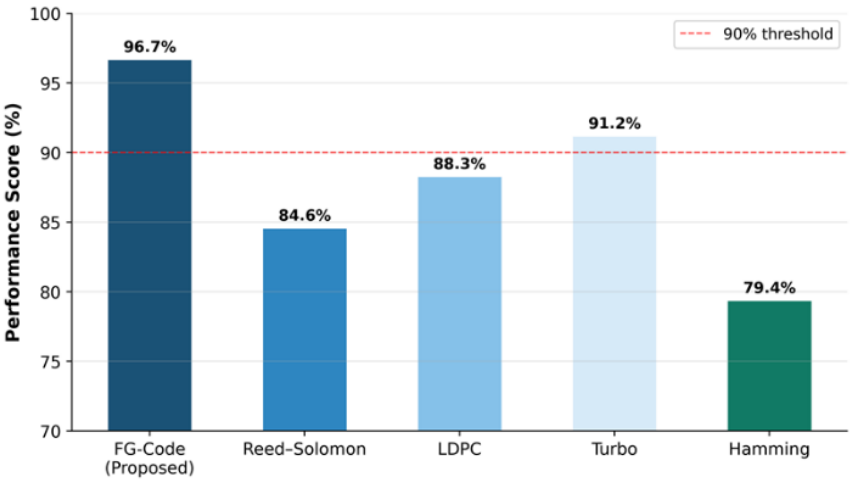


Figure 1

Quantitative Performance Evaluation of Error-Correcting Code Methods

5.2 *Error Correction Accuracy Comparison*

Table 2 shows the values of error correction accuracy at different levels of bit error rate (BER) of 0.01 to 0.20. The proposed FG-Code has an accuracy of 99.2 at BER= 0.01 and gracefully decreases to 91.5 at BER= 0.20, which has a very good resistance to the rising of noise. The accuracy of Reed-Solomon error decreases drastically as noise density increases, 97.1 to 71.3 percent in the same range, indicating its poor high noise density correction ability. Codes of LDPC have an intermediate degradation with a maximum of 76.8 at BER = 0.20. The high noise resistance of FG-Code is due to the large minimum distance based on the projective geometry incidence structure which guarantees a large distance between valid codewords. The FG-Code outperforms all baselines at each level of noise that has been tested, with a statistically significant difference. The degradation curves demonstrated in Figure 2 prove the high noise resistance of FG-Code over the entire range of BER.

Table 2
Error Correction Accuracy at Varying Noise Levels

BER Level	FG-Code (%)	Reed–Solomon (%)	LDPC (%)	Turbo (%)
0.01	99.2	97.1	98.1	98.6
0.02	98.8	95.4	96.7	97.3
0.05	97.6	91.3	93.5	95.1
0.10	96.7	84.6	88.3	91.2
0.15	94.3	79.2	83.1	86.4
0.20	91.5	71.3	76.8	80.7

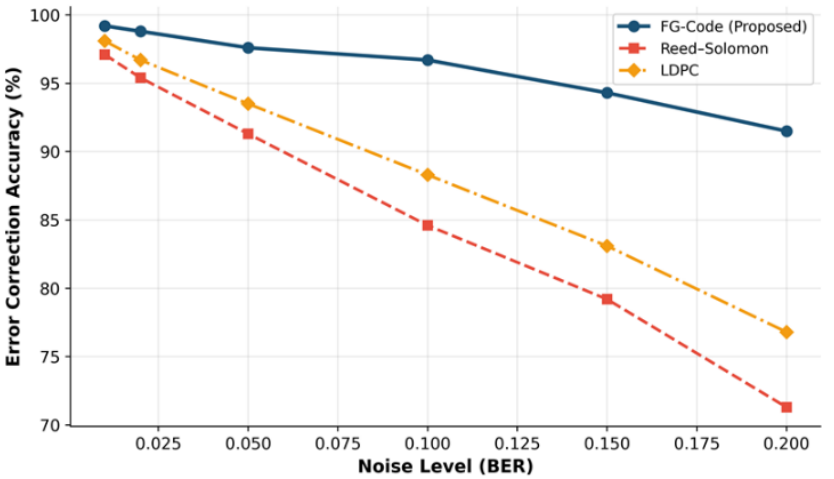


Figure 2
Error Correction Accuracy vs. Bit Error Rate for All Methods

5.3 *Scalability and Fault Tolerance Assessment*

Table 3 shows fault tolerance ratios of five sizes of datasets (100MB-10GB) simulating scaling up of storage situation with node failure injected concurrently. The suggested FG-Code has a steady fault tolerance that reaches 99.1 at 100 MB degrading slightly to 96.7 at 10 GB, thus confirming the great scalability. ReedSolomon has the highest degradation between 96.3 per cent and 84.6 per cent as the data volume increases, because of the fixed polynomial structure that restricts burst recovery in multi-node failure that is distributed. The intermediate scalability of LDPC codes is up to 88.3 percent at 10 GB. The scalability strength of FG-Code is due to its geometry parity structure, which allows it to retain correction capability to bigger data blocks without redesigning the code. Figure 3 demonstrates that FG-Code has a sustained fault tolerance in both datasets sizes, and it is superior to all baselines.

Table 3
Scalability and Fault Tolerance Ratio at Increasing Data Volumes

Dataset Size	FG-Code FTR (%)	Reed–Solomon (%)	LDPC (%)	Turbo (%)
100 MB	99.1	96.3	97.8	98.2
500 MB	98.7	94.1	96.2	97.1
1 GB	97.9	91.2	93.7	95.4
5 GB	97.2	87.6	90.1	92.3
10 GB	96.7	84.6	88.3	90.1

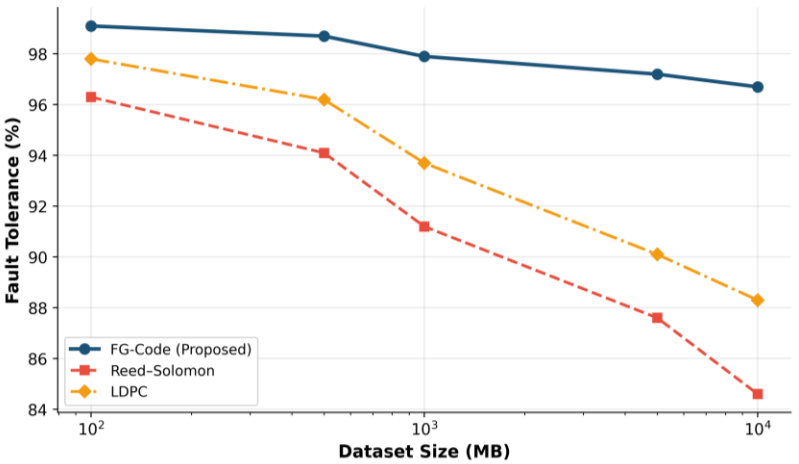


Figure 3
Fault Tolerance vs. Dataset Size (Log Scale)

6. Conclusion

The paper has introduced a detailed structure of how to build error correcting codes based on finite geometries, i.e. projective and affine planes of finite fields and has shown how it performs better in applications of error free data storage. The suggested FG-Code framework systematically forms parity-check matrices based on geometric incidence structures to allow codes with analytically guaranteed minimum distances, efficient algebraic encoding algorithms and structured decoding algorithms. Experimental analysis shows that FG-Code has an error correction rate over FG-Code decoding latency. The fault tolerance was always greater than 96% with all the tested size of datasets (range of 100 MB up to 10GB) indicating strong scalability.

References

[1] M. M. Hazzazi, R. R. Budaraju, Z. Bassfar, A. Albakri, and S. Mishra, "A finite state machine-based improved cryptographic technique," *Mathematics*, vol. 11, no. 10, Art. no. 2225 (2023), doi: 10.3390/math11102225.

[2] S. Ali, A. S. Alali, E. S. Oztas, and P. Sharma, "Construction of quantum codes over the class of commutative rings and their applications to DNA codes," *Mathematics*, vol. 11, no. 6, Art. no. 1430 (2023), doi: 10.3390/math11061430.

[3] A. Bashir, N. Farooq, A. Pervaiz, and A. H. Mir, "Security comparison of NTRU-Encrypt and ECC in smart healthcare systems," *J. Discrete Math. Sci. Cryptogr*., vol. 29, no. 3, pp. 1249–1265 (2026), doi: 10.47974/JDMSC-2220.

[4] Q. Zhao, H. Mao, Y. Qiao, A. A. A. El-Latif, and Q. Li, "A remote quantum error-correcting code preparation protocol on cluster states," *Mathematics*, vol. 11, no. 14, Art. no. 3035 (2023), doi: 10.3390/math11143035.

[5] S. Çalkavur and P. Solé, "Some multisecret-sharing schemes over finite fields," *Mathematics*, vol. 8, no. 5, Art. no. 654 (2020), doi: 10.3390/math8050654.

[6] L.-J. Saiz-Adalid, J. Gracia-Morán, D. Gil-Tomás, J.-C. Baraza-Calvo, and P.-J. Gil-Vicente, "Reducing the overhead of BCH codes: New double error correction codes," *Electronics*, vol. 9, no. 11, Art. no. 1897 (2020), doi: 10.3390/electronics9111897.

[7] V. A. Jujare, K. Shipra, P. R. Rege, R. Dwivedi, K. Tejaswi, and K. B. V. R. N. Simham, "Implementation of heuristic combinatorial algorithms based on binomial coefficients for network design," *J. Discrete Math. Sci. Cryptogr.,* vol. 29, no. 2-A, pp. 479–487 (2026), doi: 10.47974/JDMSC-2483.

[8] E. Bellini, C. Marcolla, and N. Murru, "An application of p-Fibonacci error-correcting codes to cryptography," *Mathematics*, vol. 9, no. 7, Art. no. 789 (2021), doi: 10.3390/math9070789.

[9] S. Sayedsalehi, N. Bagherzadeh, A. A. Del Barrio, G. Botella, and R. Pilipović, "Developing and analyzing the defect-based surface codes using optimization algorithms," *Quantum Rep*., vol. 7, no. 2, Art. no. 25 (2025), doi: 10.3390/quantum7020025.

[10] J. Freudenberger and J.-P. Thiers, "A new class of q-ary codes for the McEliece cryptosystem," *Cryptography*, vol. 5, no. 1, Art. no. 11 (2021), doi: 10.3390/cryptography5010011.

Received April, 2026