Secure transmission and storage of Internet of Things data

Yushang Cai *

Syracuse University, Syracuse, NY 13244, USA

* Corresponding Author Email: ycai02@syr.edu

Abstract. The Internet of Everything is the ultimate level form of the Internet of Things. The result of digitizing the objects in life is a huge amount of data that needs to be stored and transmitted. This paper will briefly introduce the IoT and the current status of data processing. Different coding methodologies, Reed Solomon (RS) codes, Low-Density Parity-Check (LDPC) codes, and Low Complexity Parity Check (LCPC) codes will also be explained in detail from their encoding and decoding aspect. Then different codes are specifically analyzed for their usage scenarios in IoT data transmission and storage based on their particular characteristics. In the experimental section, different codes are compared from different perspectives. The main comparisons are made in terms of memory usage, operation time, energy consumption per bit of encoding, and error correction performance. The results of the analysis are RS code is simple but lacks error correction, the LDPC code has good error correction but is more complex, LCPC code strikes a balance between power efficiency and performance.

Keywords: IoT, data storage & transmission, RS codes, LDPC code, LCPC code.

1. Introduction

What does life look like when all the objects in the life are not merely used but simultaneously giving feedback? The Internet of Things (IoT) will provide an answer to this question. The Internet of Things (IoT), is an emerging technology that builds a new type of ecosystem, different from the traditional one, based on devices that contain certain functions. These devices will rely on different principles, possibly through sensors that feed certain data back to the center. The different devices interact directly with the data to achieve the result of the overall environment. Thus, a digital set of brand-new environments will be built. The scope of IoT coverage will be large, ranging from small personal smart wear or smart homes to integrated agriculture or waste treatment [1, 2]. When all of these smart devices output data, the storage, and calculation of the data will have a direct impact on the operation and control of the entire system. How to process all these data becomes a critical issue. IoT has multiple data processing architectures, Grid Computing, and a distributed collection system of network connections that better utilizes existing infrastructure. Cloud Computing and Fog Computing which is used to complement Cloud Computing. Mobile-edge Computing's distributed computing environment allows computing tasks to be distributed to the edge of the data, thereby reducing latency and improving efficiency and user experience. There are also multiple architectures available such as Cloudlets, On-site Processing, and In-memory Computing [3]. Various approaches to computing ensued, with cloud computing serving as the ultimate evolution of the distributed computing model. Cloud computing is a service that provides resources related to information technology to end users [4, 5]. For data processing on cloud platforms, the main tasks can be categorized into three types, which are sensor data integration, data storage, and application support. In terms of sensor data integration, to cope with the large amount of reading and writing of user data, certain aspects of data storage will be sacrificed to achieve higher availability and flexibility. Traditional databases are difficult to provide a full range of services in response to distributed massive data processing, and can only sacrifice certain performance, such as horizontal scalability. And the vast number of databases will lack data consistency and data isolation [2]. When data storage is considered in the cloud, the perspectives of reliability, confidentiality, homomorphic, integrity, privacy, reliability, scalability, and the complexity of encryption and decryption need to be considered. Coding, or increasing the redundancy of data, is a method used to increase the reliability of data in its transmission and storage [5]. A variety of coding methods such as Reed-Solomon code and LCPC

error correction code can be used for error detection and correction [6, 7]. RS code will encode the incoming signals, decode them at the receiving end, and perform data checking, thus enabling the elimination and detection of the effects of various factors, such as noise, during the transmission of information [6]. The analysis of IoT data will reveal trends, and look for unseen details, hidden information, and specifics. This means that data, or rather massive amounts of data, have a previously unimaginable value at this moment in time. User-related data may contain personal privacy. Data interactions are caused by different types of data mining [8]. With such an onslaught, the demands for data safety are further increased. Ensuring encoding rules for data becomes even increasingly significant.

2. Different methodologies

2.1. Reed-Solomon codes

RS code, as a method of adding redundant information and thus increasing the correctness of information transmission, has become one of the more used methods due to a variety of factors [6]. The process of RS coding is relatively simple from the structural point of view, which is one of the advantages of its construction. As shown in Fig. 1, it is divided into five parts in transmission of information. The RS encoder accepts a data block as input. To construct an encoded data block, it adds a specified number of redundant bits (also known as parity bits) to the original data block. These redundant bits are computed using mathematical techniques, specifically polynomial arithmetic, and then attached to the original data. The encoded data block that results is subsequently sent via a communication channel or saved in a data storage system [6]. Communication channels that are likely to be affected by noise. In the case of storage systems, the storage process may result in data errors due to accidents. This is where the RS decoder is needed. The decoder will accept the transmitted data module and check that the parts are correct. When an error is found the data will be recalculated and recovered by correcting the algorithm which mainly consists of polynomial calculations [6]. Starting from a coding-specific point of view, the encoding of RS (n, k) will have n total characters in a single codeword. k data characters and n-k parities. Each part will have m bytes. And the maximum module length will be 2m-1 for n. And m in turn would be determined by a finite field, which means that the number of characters in the RS code would be governed by GF(2m). The RS code, as a maximum distance separable (MDS) code, has a minimum Hamming distance, and this is a metric for determining the ability to correct error or erasure. RS codes can correct

$$d_{min} = n - k + 1 \tag{1}$$

Up to $\frac{(n-k)}{2}$ erroneous symbols and corrects up to (n-k) erasures. When symbol errors or erasures exceeding this number appear in the message transmission, the RS code will not be corrected correctly [9]. The encoding of the RS code relies primarily on the RS code generation matrix G. There would be

$$G = \left(\begin{bmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_n \\ \vdots & \vdots & \vdots & \vdots \\ a_1^{k-1} & a_2^{k-1} & \cdots & a_n^{k-1} \end{bmatrix} \right) \tag{2}$$

K rows and n columns. The codeword that is considered to be the input message is of length k and is denoted m. Marked c is the encoded output. The formula for the calculation is the decoding part is

$$mG = c \tag{3}$$

Similar. When the received matrix is of length $1 \times k$, the RS generation matrix will be selected to generate a new $k \times k$ matrix with the corresponding k length. The final message m obtained at the beginning of the output will be equal to c multiplied by the inverse matrix of the new matrix Thus the

$$m = cG^{-1} \tag{4}$$

Initial signal will be restored. RS coding overall is a very powerful coding technique. Because it relies on the Galois Field (finite field) algorithm, its complexity is highly correlated with the mode and method of computing the Galois Field. Galois Field's addition calculations will be relatively simple in comparison and can be realized through XOR. Multiplication, however, will require more complex and advanced algorithms for its computation. The main thing is that polynomials are computed for different Galois Fields depending on whether m is prime or a power of 2. When m is a power of 2, there are plenty of algorithms showing that the encoding part of RS can be simplified thus making the computation faster [10].

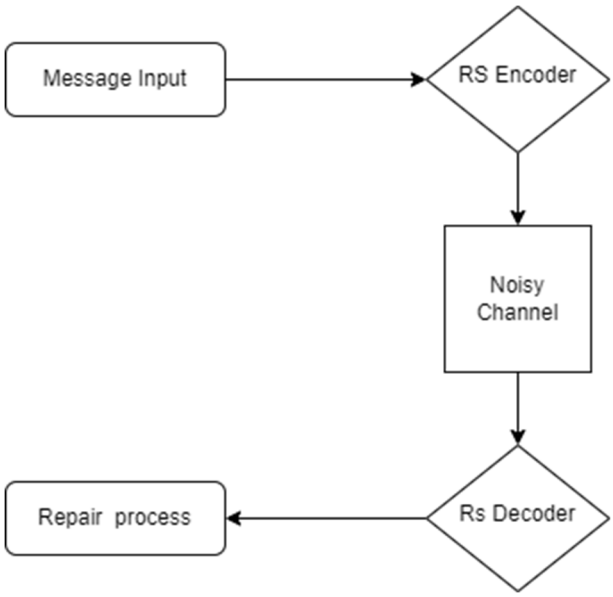

Figure 1. Process of RS code (Picture credit: Original)

2.2. Low-Density Parity-Check (LDPC) codes

LDPC (Low-Density Parity-Check) is a linear error correction coding used to detect and correct errors in data transmission in digital communications and data storage. It is a powerful error correction technique typically used to correct bit errors caused by noise, interference, or other channel problems during data transmission. LDPC coding includes several key features, the first is a sparse parity check matrix. LDPC encoding uses a sparse parity check matrix to describe the encoding rules. Most of the elements of this matrix are 0s and only a few are 1s. This sparsity makes LDPC encoding very efficient for storage and transmission. The second is the Tanner Graph, which has shown in Fig. 2 as an example. LDPC encoding can be visualized using a Tanner graph. This graph consists of a set of variable nodes and a set of check nodes representing the encoding rules. Tanner graph helps to understand how LDPC encoding works. The Tanner diagram will be based on the parity check matrix. Because of the low density of its parity check matrix, in other words, less distribution of '1's in the matrix, it enables LDPC coding to be more efficient in high-rate data transmission and storage [11]. The encoding and decoding process of LDPC is more complex. For the encoding process, the main thing is to pass the input of the message through the parity check matrix to generate the data of the check bits. The whole block of data is generated by combining it with the original data. This data block can be used for transmission and storage [12]. The decoding process may vary in both process

and computation depending on the algorithm. The Tanner diagram is first generated from the parity check matrix used in the encoding. Different parts of the data block will be assigned different probability values. The data nodes and check nodes will be transmitted through the Tanner diagram shown for several transmission iterations [7]. Thus, the number of iterations is reached or the error correction requirement is satisfied. Finally, the result of decoding is output. Similar to RS codes, LDPC codes have new coding methods that are updated from the original. LDPC convolutional codes, for example, are in the process of temporally integrating blocks that were originally multiple LDPC codes [13].

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

Figure 2. Tanner diagram example for LDPC [11]

2.3. Low Complexity Parity Check (LCPC) codes

To reduce the complexity of encoding and reduce the amount of computation involved in decoding. Low Complexity Parity Check (LCPC) codes have been proposed by some researchers. Compared to RS and LDPC codes, LCPC has smaller coding and decoding requirements because it does not involve an iterative process. It can be applied to very low memory for low-complexity scenarios. This is very attractive for IoT scenarios with low latency and high computational feedback rates [7]. The LCPC code divides the whole data into blocks of the same length, (n, k). Where n is the code word length while k is the data length. The encoding process will generate redundant parity bits through the data bits. The number of parties is $n - k$. As shown is Fig. 3, it is an example for the G and H matrix for LCPC (8, 3).

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

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

Figure 3. Encoding matrix G and decoding check matric H for LCPC (8, 3) [7]

Because the rows in the H matrix are orthogonal to those in the G matrix. This means that the internal product of G and H will be 0. For the received code word CDTi, the product of H and its will be 0 when there are no errors in the internal message transmission:

$$H \times CD_{Ti}^{T} = 0 \tag{5}$$

By the same token, the result will not be 0 when an error occurs. and for a particular LCPC code, the type and location of the error can be known when the number of errors is known [7]. Such as Table 1, it is an example table for checking. If we know the error pattern which is called a syndrome vector. We can figure out the error pattern in the data from different condition tables. Then the entire decoding method would be shown in Fig. 4. Intermediate process because it involves binary addition.

so, it can be realized simply by the process of XOR. This also implements LCPC codes less complex and thus increases the speed of operation.

Table 1. Error pattern & syndrome vector for single bit error of LCPC (9, 4) code [7].

Error pattern EP	Syndrome vector SY
000000000	00000
000000001	00001
000000010	00010
000000100	00100
000001000	01000
000010000	10000
000100000	10111
001000000	11011
010000000	11101
100000000	11110

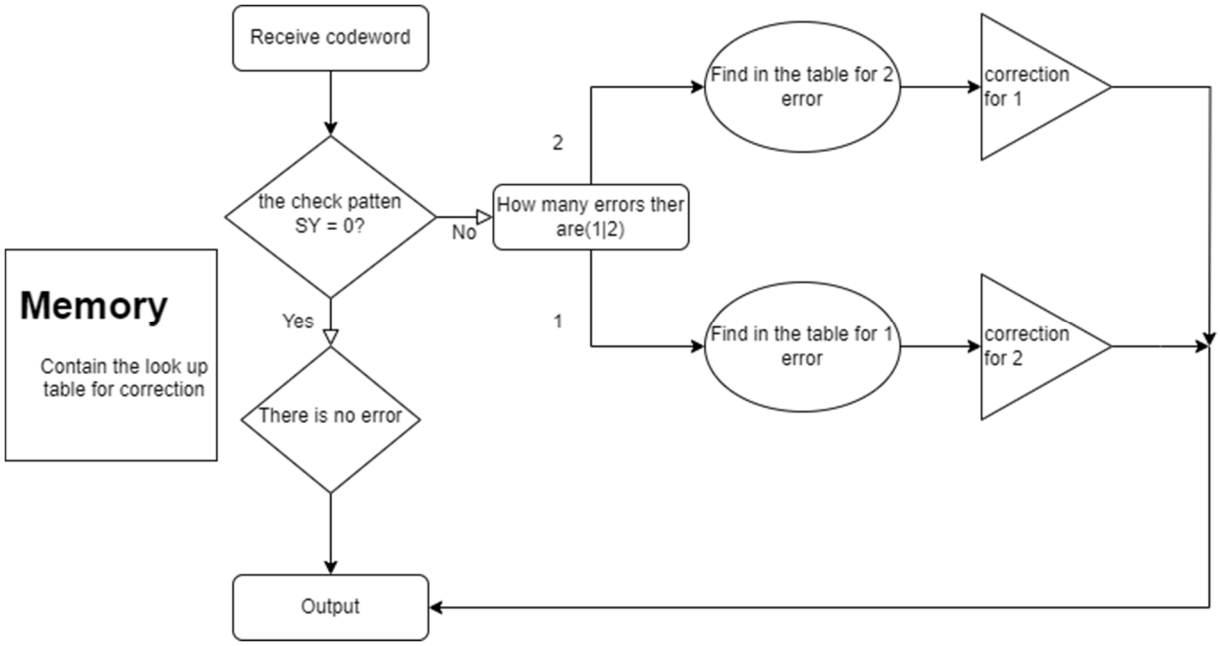

Figure 4. Decoding logic for LCPC code [7]

3. Requirements for secure transmission and storage of IoT data

When large systems such as smart grids, healthcare, and agriculture are being rectified into the IoT, it means that a lot of data integration is involved. The approach to data storage and processing involved in the introduction is the more traditional cloud-based model of the IoT. For different scenarios, it is possible that cloud services may not be able to fulfill all the important needs. For example, if the cloud accepts and stores large amounts of data, it requires a lot of hardware storage space to support it. At the same time, the accumulation of large amounts of data can make the servers accommodate more sensitive data, thereby increasing the likelihood of leakage [5, 14, 15]. These are major issues that cannot be ignored in data processing. In this case, decentralization, Mobile-edge Computing, or Fog Computing will become a more secure solution [3, 14]. However, data protection for marginalized and distributed storage becomes problematic. Blockchain is the protection platform that was created to deal with this problem. In this model, various data will be stored in Distributed Hash Tables (DHTs). The blockchain built by the user will be a lock, thus protecting access to the data. The blockchain will contain a pointer to the address where the data is stored. When an access request is granted by the blockchain, the user will be able to find the data through the pointer [14, 15].

For blockchain, because of its restrictions on user access to data, the security technology aspect is mainly encrypted through cryptography-based secret keys. To maintain the basic normal interaction of data, encryption protocols, and error-correcting codes are ways that can reduce uncontrollable risks. Data assurance is something that can be analyzed from multiple perspectives, including Reliability, Scalability, Accountability, Availability, Confidentiality, Integrity, and Privacy [5]. It is beyond the scope of this article to discuss encryption. There are a lot of detailed tables for the comparison of different methods, so this article won't be released due to space issues. Detail information can be found in [5]. For coding, Reliability, and Availability will be the biggest perspectives on what it can do to help in data storage and transfer [5]. This leads to the fact that different codes are suitable for different scenarios due to their specific properties.

4. Different coding methods apply to IoT data analysis.

4.1. Reed-Solomon codes

It must be stated at the very beginning that the relevant coding discussed in this paper is broad in range, and that the same code can be used in many ways to improve the computing architecture and results through algorithmic improvements and so on. This paper will only be able to analyze some of the hypothetical scenarios. From the internal structure of the hardware implementation, the encoder of the RS code can be described as a shift register. Different operations are performed on the input signals to output a result for transmission [6]. Relatively speaking it's still straightforward. The decoder is more complicated. The whole brief structure is shown in Fig. 5. The accepted data will be computed as syndromes and then checked for errors. When an error occurs Barlekamp algorithms will be used for calculating the polynomial of the error. The Chein search algorithm is used for calculating the root of this polynomial. Then Forney's algorithm is used to calculate the error location. This data will be integrated and then error correction will be done through an error corrector [6]. Architecturally, RS code is more concise to implement and has good competition in terms of error correction performance. So, it is suitable for automation control systems in the home. It can be used to correct the errors caused by scratches dust and other disturbances in data storage [6]. The Fast Fourier Transform (FFT) and its Inverse Transform (IFFT) can be used to accelerate the matrix multiplication and inversion of finite-domain-based RS codes during computation. This will speed up the encoding and decoding of RS codes to some extent [16, 17]. RS codes can also be realized by other criteria for example, without using syndrome for computation, the complexity will be increased, which is not favorable to the application of RS codes for data transmission and storage in reality [17].

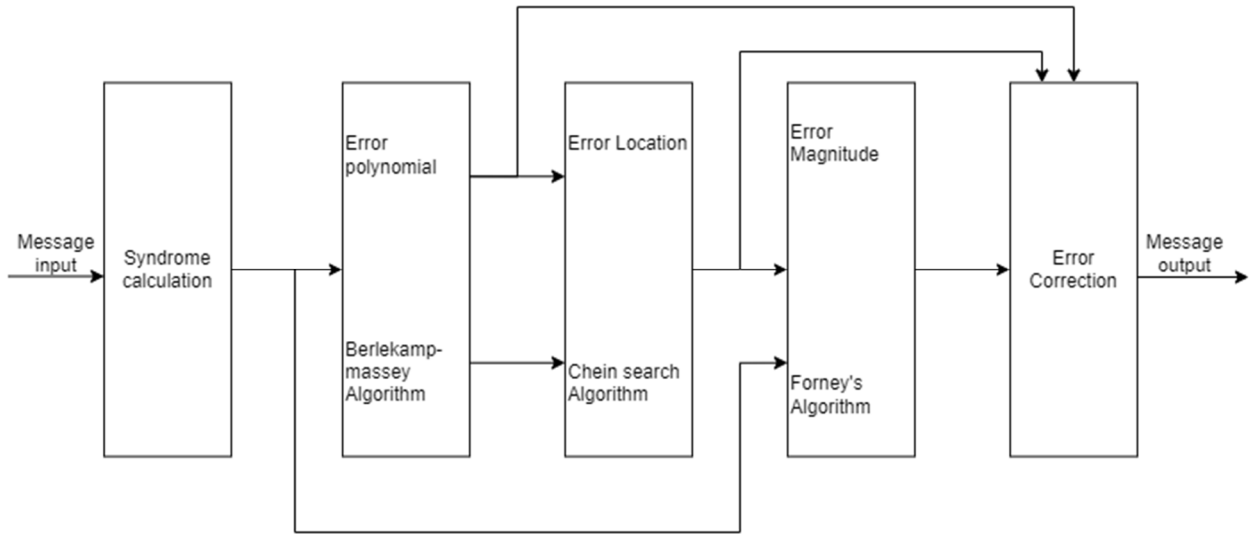

Figure 5. Overall structure for RS decoder [6]

4.2. Low-Density Parity-Check (LDPC) codes

In terms of error correction, LDPC codes are one of the best-performing error correction codes, which are close to the performance of the Shannon limit [7, 18]. Because of its high performance, its main application scenario is in integrated circuits. They are used in large quantities in custom integrated circuits (ICs), programmable gate arrays (FPGAs), or GPU devices [11, 18]. However, the computational complexity of LDPC is a barrier to accelerating op-computing. Modern LDPC codecs are mainly suitable for scenarios and communication standards with high throughput requirements. This is contrary to the large demand for low hardware requirements and fast processing times in the IoT. Although some research has been done to improve the original LDPC codes and further optimize the codecs so that the LDPC codes can be effectively stored in the restricted low-power system RAM, the hardware construction and implementation of the specific codecs still need further investigation [18]. Because of this, the application scope of LDPC code in IoT will be limited. The main focus is on the processing of big data, such as remote communication or high-tech smart electronic products, which have a high demand for data reliability. Smaller scenarios are mainly battery-controlled on low-power devices.

4.3. Low Complexity Parity Check (LCPC) codes

The advantage of LCPC codes is that it has lower complexity and lower storage requirements compared to RS codes and LDPC codes. This is due to the fact that no iterative computation is required in the decoding stage, which greatly reduces the computation process and thus the complexity of the application. As stated in the Methodology Introduction, the processing logic is very clear for the error correction part of the LCPC code. And the lookup tables of each part can be stored directly in memory. This allows the data interactions in the error correction process to be analyzed and designed straightforwardly, thus simplifying the application [7]. For specific applications in IoT, LCPC codes can be used for low-power consumption devices or low bandwidth usage scenarios. It can also be used for sensor nodes and other similar environments where data is widely distributed and fragmented. The use of LDPC codes in these cases would not be particularly effective.

5. Performance evaluation

This experimental study will first focus on the performance comparison between RS codes and LDPC codes. The perspective of the comparison will start with the memory consumption aspect. Focusing first on Fig. 6, the horizontal coordinate shows the different parity matrices used for the experiment and the vertical coordinate shows the memory usage. It can be seen from the data that the base LDPC code has a very high demand in terms of encoding memory usage. The computation process requires iteration which makes the memory usage exponentially higher than other codes. Precisely because the memory requirements are so large, this puts a limit on the large-scale applicability of basic LDPC codes. QC-LDPC is not significantly different from RS code in terms of memory usage. Each has strengths and weaknesses when using different generating matrices and corresponding block codes. After discussing the use of encoded memory, it is also necessary to compare the use of time. In Fig. 7, the vertical coordinate is this time replaced with time usage for a single block. As can be seen from Fig. 7, the improved LDPC code from basic LDPC consumes more time than the RS code in most cases. However, the QC-LDPC code is able to significantly reduce the time consumption and thus increase the efficiency. Next, in terms of hardware construction, it is also necessary to consider the energy consumption required for the encoding process. This will also be directly linked to the cost of encoding. In Fig. 8, it shows the relationship between code length and the encoding energy consumption per bit. It can be learned from Fig. 8 that an increase in code length will increase the energy consumption per bit encoded. Meanwhile, RS codes will be superior to LDPC codes in this aspect, but consume more energy than QC-LDPC codes. Finally, in this part, there is also a comparison of the error correction performance. As can be seen in the representation in Fig. 9, the signal-to-noise ratio of the RS code is nearly twice that of the LDPC code for nearly the same

BER. This means that RS codes require nearly twice the energy consumption for the same error correction performance. The final part then experimentally evaluates the performance of LCPC codes. The main comparison is on the error correction of LCPC codes. What is still listed in Fig. 10 is the comparison of BER and signal-to-noise ratio. Fig. 10 clearly show that LCPC (7,3) has better error correction capability than other LCPC schemes with the same power. LCPC (7,3) will reduce the power consumption by 25% compared to the RS code with the same error correction capability. When the technique of RS is replaced then the results will be different. As shown in Fig. 11, when the RS code becomes RS (15,11), in high power consumption scenarios the RS code will have little difference in error correction capability with the LCPC code. However, in low power consumption scenarios, the error correction capability of LCPC codes is better. LCPC codes will also be compared with LDPC codes. It will be much more complicated at this point. The different decoding architectures of LDPC will seriously affect its relevant performance. According to Fig. 12, it can be concluded that the decoding performance and power consumption of LDPC are worse than that of LCPC code in some configurations. However, other structures may be roughly the same or in some cases better than LCPC codes.

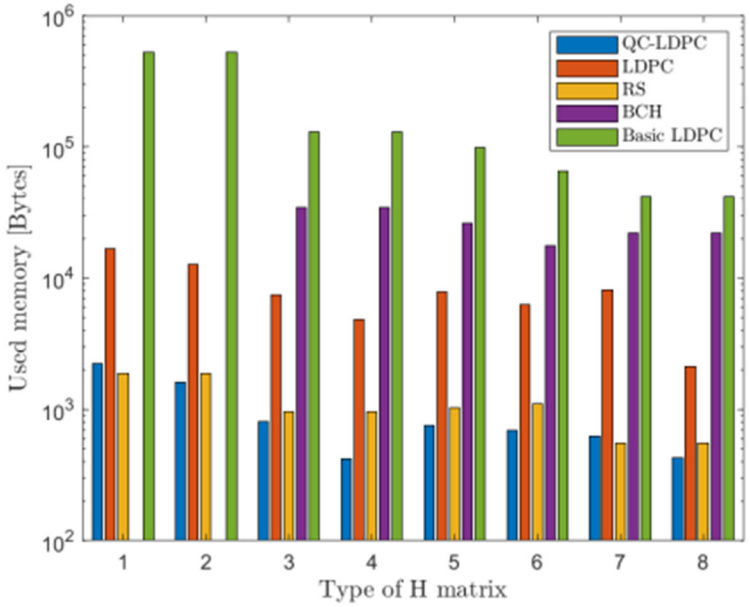

Figure 6. Memory utilization for encoder [18]

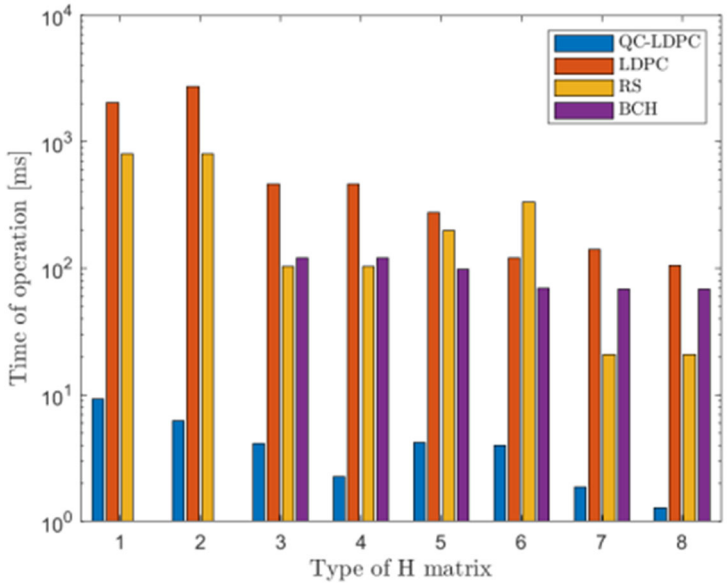

Figure 7. Coding time consumption for a single block [18]

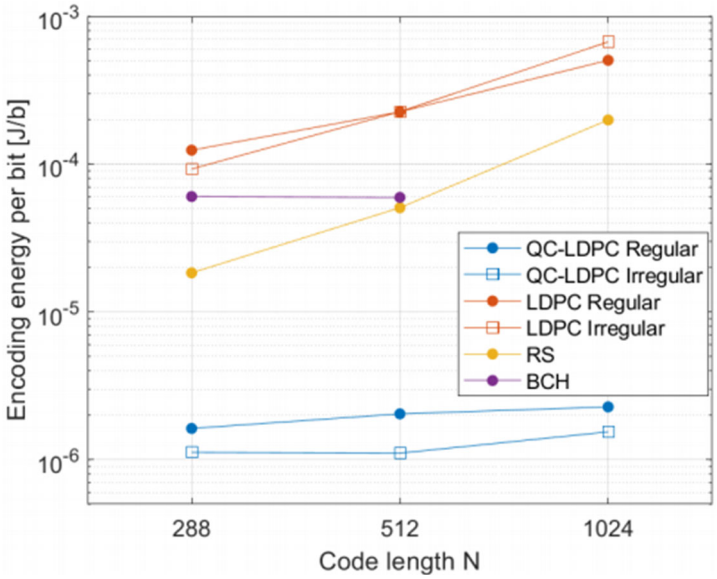

Figure 8. Code length vs. encoding energy consumption per bit [18]

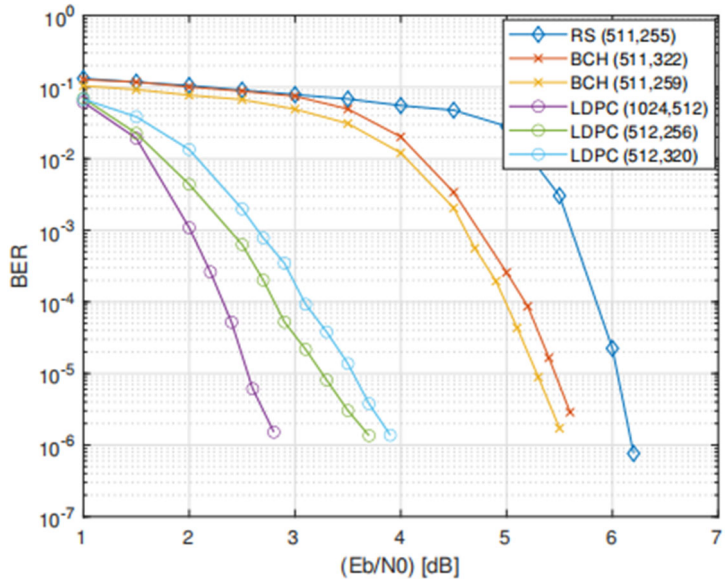

Figure 9. Bit error rate curve vs. the effective signal-to-noise ratio [18]

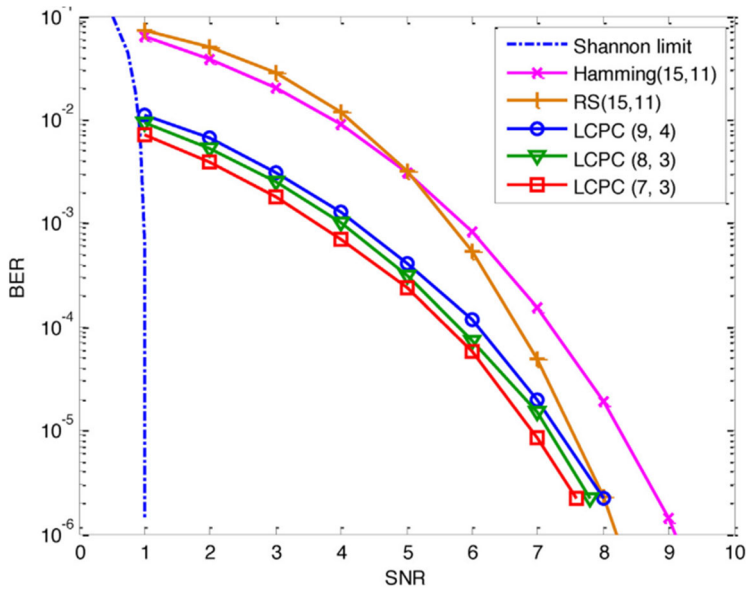

Figure 10. BER vs SNR for LCPC (9, 4) code and other codes [7]

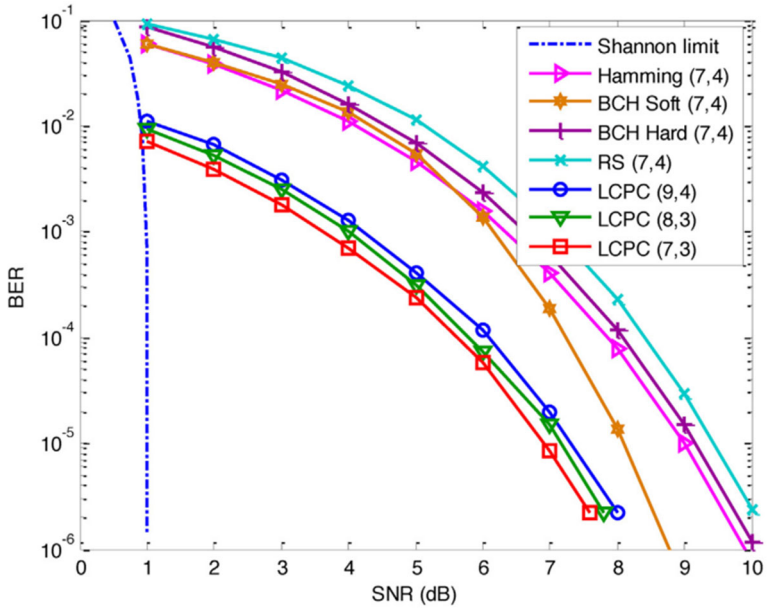

Figure 11. BER vs SNR for LCPC (9, 4) code and RS (15,11) code [7]

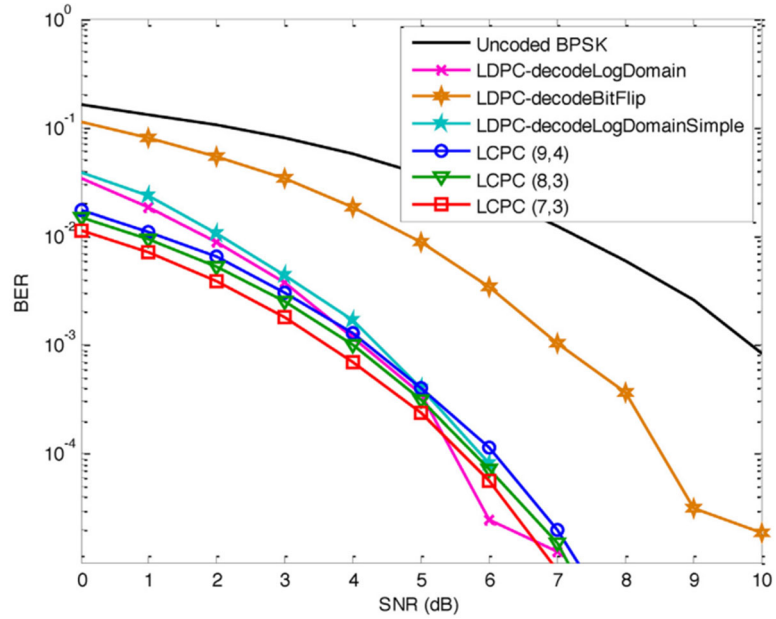

Figure 12. BER vs SNR for LCPC code and LDPC code [7]

6. Conclusion

In this paper, different coding approaches are analyzed and discussed in the setting of IoT data processing. What can be derived is that RS code because of its simpler structure and process will be the right choice for simple data processing, such as smart home and other usage scenarios. However, its error correction ability and power saving are hardly the best without further support from algorithms. LDPC codes are the opposite, with algorithms that support high-performance error correction and large data processing. However, the cumbersome architecture will also limit its application when data is decentralized and costs need to be controlled. Although improvements have been made to reduce the complexity, the memory requirements still need to be investigated. LCPC codes, on the other hand, can be seen as a balance of the two. That is, it provides relatively low consumption and guarantees acceptable processing performance. RS codes and LDPC codes have different architectures, algorithms, and technical support. Many of them will drastically improve the related practical application results. This paper explores a very small part of it in a rather superficial

way, and some many more settings and conditions can be analyzed and explored. Meanwhile, coding is only a tiny part of the IoT used to protect transmitted and stored data. Other data architecture changes or cryptographic encryption methods can be used in combination.

References

[1] N. Lal, S. Qamar, S. Agarwal, A. Kumar Agarwal, and S. Singh Verma, Internet of Things: Applications for Sustainable Development, 1st ed. Boca Raton: Chapman and Hall/CRC, 2023. doi: 10.1201/9781003226888.

[2] Lihong Jiang, Li Da Xu, Hongming Cai, Zuhai Jiang, Fenglin Bu, and Boyi Xu, "An IoT-Oriented Data Storage Framework in Cloud Computing Platform," IEEE Trans. Ind. Inf., vol. 10, no. 2, pp. 1443 – 1451, May 2014, doi: 10.1109/TII.2014. 2306384.

[3] Shishir K. Shandilya, Soon Ae Chun, and Smita Shandilya, Internet of Things Security: Fundamentals, Techniques, and Applications. in River Publishers Series in Information Science and Technology. Aalborg: River Publishers, 2018. Accessed: Sep. 28, 2023.

[4] S. Saxena and A. K. Pradhan, Eds., Internet of Things: Security and Privacy in Cyberspace. in Transactions on Computer Systems and Networks. Singapore: Springer Nature Singapore, 2022. doi: 10.1007/978 - 981 - 19 – 1585 - 7.

[5] N. Chervyakov, M. Babenko, A. Tchernykh, N. Kucherov, V. Miranda-López, and J. M. Cortés-Mendoza, "AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security," Future Generation Computer Systems, vol. 92, pp. 1080–1092, Mar. 2019, doi: 10.1016/j.future. 2017. 09. 061.

[6] I. A. Shah, F. A. Malik, and S. A. Ahmad, "Enhancing security in IoT based Home automation using Reed Solomon Codes," in 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India: IEEE, Mar. 2016, pp. 1639 – 1642. doi: 10.1109/WiSPNET.2016.7566417.

[7] S. A. Alabady, M. F. Mohd Salleh, and F. Al-Turjman, "LCPC error correction code for IoT applications," Sustainable Cities and Society, vol. 42, pp. 663 – 673, Oct. 2018, doi: 10.1016/j.scs.2018.01.036.

[8] M. Marjani et al., "Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges," IEEE Access, vol. 5, pp. 5247 – 5261, 2017, doi: 10.1109/ACCESS. 2017. 2689040.

[9] J. Xu, "Improved least significant bit algorithm based on RS-code," in Second International Conference on Advanced Algorithms and Signal Image Processing (AASIP 2022), K. Subramaniyam, Ed., Hulun Buir, China: SPIE, Nov. 2022, p. 43. doi: 10.1117/12. 2659479.

[10] J. S. Plank, "The RAID-6 Liberation Codes," presented at the FAST '08: 6th USENIX Conference on File and Storage Technologies, Feb. 2008. [Online]. Available: https://www.usenix.org/legacy/event/fast08/tech/full_papers/plank/plank_html/index.html

[11] A. Mondal, S. Thatimattala, V. K. Yalamaddi, and S. S. Garani, "Efficient Coding Architectures for Reed–Solomon and Low-Density Parity-Check Decoders for Magnetic and Other Data Storage Systems," IEEE Trans. Magn., vol. 54, no. 2, pp. 1 – 15, Feb. 2018, doi: 10.1109/TMAG.2017.2778053.

[12] Edgar Martinez-moro, Algebraic Geometry Modeling in Information Theory, no. v. 8. in Series on Coding Theory and Cryptology. Singapore: World Scientific, 2013. Accessed: Sep. 29, 2023. [Online]. Available: https://libezproxy2.syr.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=e000xna&AN=545479&site=ehost-live.

[13] D. J. Costello, D. G. M. Mitchell, P. M. Olmos, and M. Lentmaier, "Spatially Coupled Generalized LDPC Codes: Introduction and Overview," in 2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC), Hong Kong, Hong Kong: IEEE, Dec. 2018, pp. 1 – 6. doi: 10.1109/ISTC. 2018. 8625327.

[14] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for Large-Scale Internet of Things Data Storage and Protection," IEEE Trans. Serv. Comput., vol. 12, no. 5, pp. 762–771, Sep. 2019, doi: 10.1109/TSC. 2018. 2853167.

[15] C. Yu, N. Mei, C. Du, and H. Luo, "Blockchain Data Scalability and Retrieval Scheme Based on On-Chain Storage Medium for Internet of Things Data," Electronics, vol. 12, no. 6, p. 1454, Mar. 2023, doi: 10.3390/electronics12061454.

[16] J. Yang, W. Jia, Z. Gao, Z. Guo, Y. Zhou, and Z. Pan, "Cuckoo-Store Engine: A Reed–Solomon Code-Based Ledger Storage Optimization Scheme for Blockchain-Enabled IoT," Electronics, vol. 12, no. 15, p. 3328, Aug. 2023, doi: 10.3390/electronics12153328.

[17] N. Chen and Z. Yan, "Complexity Analysis of Reed-Solomon Decoding over GF (2 m) without Using Syndromes," J Wireless Com Network, vol. 2008, no. 1, p. 843634, Dec. 2008, doi: 10.1155/2008/843634.

[18] J. Hyla, W. Sułek, W. Izydorczyk, L. Dziczkowski, and W. Filipowski, "Efficient LDPC Encoder Design for IoT-Type Devices," Applied Sciences, vol. 12, no. 5, Art. no. 5, Jan. 2022, doi: 10.3390/app12052558.

Appendix. Terminology

Reliability is the ability to keep the system running when it fails because of other operations such as fault tolerance mechanisms.

Scalability is the ability for a system to increase its operational load, or to function properly as it increases its workload.

Accountability means that any role in the system has its own responsibilities and job functions.

Availability means that for any licensed parities, the relevant content of the system can be accessed under any circumstances, regardless of the system environment.

Confidentiality means that any unauthorized group does not receive, steal, or use data.

Integrity means that the data must be kept intact. It cannot be modified or deleted by any unauthorized group.

Privacy means that an individual's personal information should be protected.

Bit Error Rate (BER) is used to evaluate the probability of an error occurring in the bit stream received at the receiving end. The probability of an error occurring in how many bits of each transmitted bit. It is usually expressed as a percentage or as a decimal.

Effective Signal-to-Noise Ratio (Eb/N0, SNR) is an important parameter used to measure the performance of digital communication systems. It is the ratio of signal power to noise power, usually expressed in decibels (dB).