Journal of Emerging Computer Technologies
Research Article
Received: *2021-08-19* | ***Reviewing:*** *2021-08-19* & *2021-09-18*| ***Accepted:*** *2021-09-21* | ***Online:*** *2021-09-21* | ***Issue Date:*** *2021-12-31*

Coding Schemes for DNA Patient Record Processing to Electronic Health Records Systems

Izabela Mitreska
UIST “St. Paul the Apostle”
Ohrid, Republic of North Macedonia
izabela.mitreska@cse.uist.edu.mk
0000-0002-0848-9416

Ninoslav Marina
UIST “St. Paul the Apostle”
Ohrid, Republic of North Macedonia
ninoslav.marina@gmail.com
0000-0003-4862-0199

Natasa Paunkoska (Dimoska)
UIST “St. Paul the Apostle”
Ohrid, Republic of North Macedonia
natasa.paunkoska@uist.edu.mk
0000-0001-9639-2552

***Abstract*—Lately, electronic health record (EHR) systems became very popular in medical technology. The main aim of such systems is to perform a digital version of a patient’s paper chart. EHRs are real-time, patient-centered records that make information available instantly to authorized users. One critical patient record is the DNA sequence, which should be processed and stored in the EHR without any modifications. Therefore, in this paper, we focus on how DNA sequence can be reliably processed to EHR systems. By introducing coding technique on top of the information we implemented the wanted security. We consider and analyze two coding schemes, the Hamming code and Reed-Solomon, on the same data sample. The results are summarized and compared by error detection and error correction values. The final outputs show that Reed-Solomon coding scheme outperforms the Hamming code scheme for reliably and securely processing the DNA record to the EHR.**

Keywords— Electronic Health Records, encoding, decoding, Hamming code, Reed-Solomon code.

I. INTRODUCTION

The constant emergence of new technologies on a global scale introduced the digitalization of healthcare services by implementing different Electronic Health Records (EHR) systems. EHR systems are electronic versions of patients' medical and treatment card history that improve health surveillance and clinical decision making. The availability of complete medical information allows physicians to distinguish chronically ill patients and identify the proper diagnose intended to provide medical treatment. Early intervention of health-related issues is fundamental for the effective treatment and avoidance of further medical complications. Mitreska et al. state in [1] that by granting adequate diagnoses and treatments, medical personnel gain the opportunity to safeguard people's lives in an effective and timely manner.

Nowadays there is a common implementation practice of EHR systems in patient's data management as shown in Fig. 1, i.e. introducing health tracking, diagnoses, different applied therapies, physicians’ reports, information for deoxyribonucleic acid (DNA), etc., to improve the healthcare process. Hence, this type of system can be applied to create flexible architectures that facilitate healthcare structure interoperability. The main characteristics of EHR systems are proprietary data flow formats and encoding schemes, which hinder the possibility of sharing data in a standard format. For example, data from a hospital that offers cancer treatments, which is a source for further data processing in this situation, should be extracted and mapped to the EHR system of other department or hospital that offer different service as depicted in Fig 2. This means that in order to realize successful general implementation, a special emphasis must be put on the sensitive data transfer path or the information processing among the diverse entities.

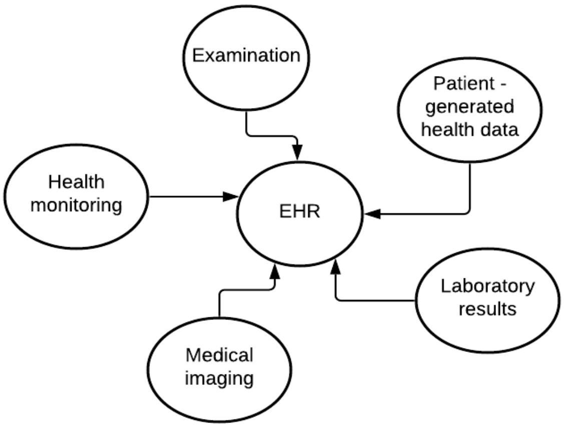

Fig 1. EHR systems

Therefore, this paper chose one sensitive patient information, a DNA record, to be processed to the EHR system. For clarification, DNA is a source of patient information and nucleotides consisting of five-sided sugar, a phosphate group, and a base. There are four different types of nucleotides, each defined by a specific base: A (Adenine), C (Cytosine), G (Guanine) and T (Thymine) [2]. The nucleotides depend on the DNA sequencing order, which indicates how important the way of processing the critical data among the entities is. This paper focuses on how the DNA record can be transferred reliably to the final location; thus, we will skip further discussion for the DNA sequencing concept.

Coding schemes are popular methods in todays' networks for sending information successfully to the end destination. There are various methods invented until now, but the general idea behind all diverse schemes is adding some predefined redundancy to the useful part of information in order to prevent the data from dealing with some errors and modifications that can appear during the transmission process. In our paper, we focus on two different coding methods for processing the data, namely a Hamming code and the Reed-Solomon code [3]. Both schemes, through examples, are applied to the same piece of DNA record and

Cite (APA): Mitreska, I., Marina, N., Paunkoska (Dimoska), N. (2021). Coding Schemes for DNA Patient Record Processing to Electronic Health Records Systems. *Journal of Emerging Computer Technologies*, 1(2), 49-54.

accordingly, the number of error detection and corrections are calculated and compared. The operation mentioned previously shows which code scheme performs better for DNA record processing reliably and securely to the EHRs.

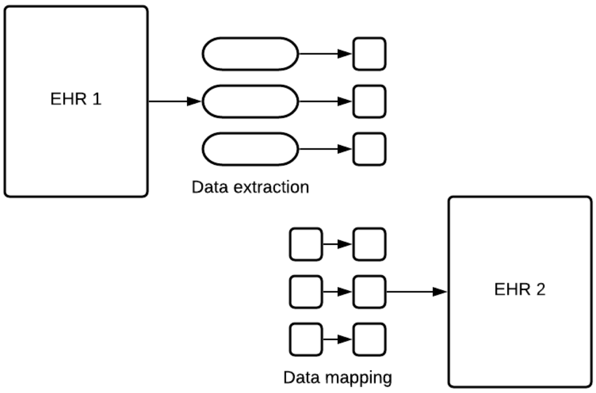

Fig. 2 EHR data processing

Nevertheless, this paper is organized as follows: Section II surveys the concepts related to EHR systems and explains different techniques for patient data processing without significant modifications. Section III and IV, respectively, explain how encoding and decoding work with Hamming and Reed-Solomon code for patient data transmission. Section V summarizes and compares the results from the previous sections, and section VI concludes the paper.

II. Related Concepts

Reliability and security are fundamental concerns in any healthcare system. Different researches have proposed many security reference architectures. One of them is given in [4] by proposing this kind of architecture that can be seen as a base point to study security threats and their characteristics. Paper [5] elaborates on specific reliable architecture for patient data and healthcare services management. Consequently, the benefits of existing EHR systems became better understood, and the performance gains associated with EHR adoption were clarified. [5]

Another way of looking at security in EHR systems is the attempt to protect physician's services and patient data from various attacks done by third parties. Paper [6] defines different security aspects related to authorization, authentication, encryption and access control for EHR systems. Issues around data security, trustworthiness and privacy today are under greater focus than ever before. As a result, many techniques for data protection have been developed over the past 20 years [6]. Reference [7] provides a broad perspective about the variety of research that can contribute to developing effective and efficient data protection technologies.

Other aims for the creation of the EHR are transparency, openness, reliability, performance and scalability. Implementing such systems with the objectives mentioned above is elaborated in [8]. The first EHR systems were implemented in 2001 using the concept of paper-based records. Today EHR systems are quicker, more secure and more accurate than the traditional paper-based records because they consider the difference in age, gender, job title, previous computer experience and education levels [9].

The increased rate of adoption of EHR systems at hospitals rather than paper-based records demonstrates the efficiency of how the patients are treated. The advantages are enormous because hospitals want to deliver quality healthcare for their patients without severe cost overruns [10]. The usage of the EHR systems greatly increases the precision and comprehensiveness of medical data, which will enhance standards and disease prevention capabilities. Databases consisting of medical records make data more easily shareable between providers and organizations [11].

Many systems designs, including EHR systems, have been proposed to address information availability challenges. Considerations for security in protecting data are mostly ad hoc and patch efforts which may not be well thought out as part of an overall security architecture. Researchers in [12] show that attribute-based authorization can be a critical architectural component for protecting healthcare systems and their users from insider attacks. Smart healthcare services are a great boon and are dominantly used by patients, doctors and other healthcare providers. Since most data is stored in cloud servers, there is an imminent need to safeguard them from unauthorized access. Existing smart health solutions, i.e. e-health cloud preserving cryptographic and non-cryptography mechanisms, provide a privacy aspect in the cloud. The evolution of such security mechanism can make health care data more secure and sustainable [13]. The intent of chapter [14] has been to outline how EHR systems work and how different mechanisms support such systems' operations to avoid security issues. Managing and storing the Big Data in EHR systems is a big challenge. Therefore, the paper [15] provides an overview of all methods used in order to achieve data security in different systems. Different coding techniques, encryption algorithms and classifications were done to determine which security method is adequate to deal with what kind of attack. A well-known group of popular error-correcting codes is also considered for DNA record protection in EHRs. Some researches can be found in references [18-20].

The inventions of such systems improve the worth and effectiveness of healthcare. The high satisfaction of medical data collected in such systems results in the functionalities available for prescribing drugs [16]. In [17], a novel method was proposed, which was used to construct and securely store shadows of medical images. The experimental results demonstrate that using (7, 4) Hamming code gives a more desirable blurring effect than using (15, 17) Hamming, because the scheme that was proposed by the authors runs much faster at low computational costs which is suitable for mobile devices or small size hospitals or clinics. All of the concepts mentioned above explain different ways for patient data security. But, through our investigation, we gain knowledge that we need to compare two coding techniques with different performances, as a novel method, to give a better solution for patients' data security in EHR systems. This is why we implemented the Hamming code and Reed-Solomon code for DNA data transmission as critical patient records that should be processed and stored in the EHR without any single modification. The constructions of the two codes are elaborated in [3], [18-20].

III. Hamming Code

This section uses a concrete example to explain how the Hamming code encodes and decodes DNA records. Then, it calculates the number of errors detected and corrected during the data transfer process to other EHR systems.

A. Hamming Code applied to DNA record

We considered the Hamming (31, 26) code to encode the data consisted of the strings **s**=*hello*. This string is taken for simplicity to represent the example, but in reality, it should be seen as a sensitive DNA record. For the encoding process of this code, we need to construct a particular generator matrix. For this purpose, we use the matrix *A* with dimension 5x26 given in equation (1) and an identity matrix with dimension 26x26. Thus the *generator matrix* G with dimension 26x31 is obtained by taking $G = [\, I_{26} \;\; -A \,]$. Additionally, we construct the *parity-check* matrix $\boldsymbol{H}$ with dimension 31x5 by taking $H = \begin{bmatrix} A \\ I_5 \end{bmatrix}$, where A is the matrix from (1) and I_5 identity matrix with dimension 5x5.

In the supplementary information of [3], the Python code used for encoding and decoding DNA data storage was given. To understand these two procedures, we will analyze and discuss them more thoroughly in this paper.

B. Encoding data string to DNA record using Hamming (31, 26) code

This section demonstrates all steps needed for data encoding using the Hamming (31, 26) code to save the DNA record and transmit it to the EHR system safely to the final destination.

Step 1. Firstly, the string **s** = *hello* is converted to numerical string. The conversion procedure is following: the *UTF-8* encoding is used in order to convert all of the letters from the string **s** to *ASCII* symbols. Those numerical symbols are then converted to *base 4* numbers and they are concatenated. The codes in this paper are applied to quaternary digits {0, 1, 2, 3}, known as *quads*. So, the resulting message of 20 quads is denoted as

$$\boldsymbol{m} = 12201211123012301233.$$

The process of string-number conversion in details can be seen in Table 1.

Step 2. In order to count all unique symbols in one string we need to add a *cyclic redundancy check* (CRC) that contains the Secure Hash Algorithm known as SHA-256. This is put into *base 4* form and the 6 right-most quads are taken. The result of this step is denoted by

$$\boldsymbol{h} = 110213.$$

After that the outputs $\boldsymbol{m}$ to $\boldsymbol{h}$ are concatenated obtaining message with 26 quads

$$\boldsymbol{a} = 12201211123012301233110213.$$

letter	*ASCII*	*base 4*
h	*104*	*1220*
e	*101*	*1211*
l	*108*	*1230*
l	*108*	*1230*
o	*111*	*1233*

***Table 1**: Step 1 –Encoding data to DNA using Ham (31, 26) code*

Step 3. The next step is when the generator matrix $\boldsymbol{G}$ comes into play which is used in order to encode the message $\boldsymbol{a}$. Hence, by multiplying $\boldsymbol{a}$ and $\boldsymbol{G}$ we get the following result with 31 quads in total

$$\boldsymbol{aG} = \boldsymbol{b} = 1220121112301230123311021321131.$$

Step 4. To make sure that errors can be detected and corrected we need to know that not all words are codewords. Hence, we should add a parity check quad $\boldsymbol{p}$ which is 3 in our case (by performing on $\boldsymbol{b}$ modulo operation 4), resulting with the final codeword with 32 quads presented as

$$\boldsymbol{c} = 31220121112301230123311021321131.$$

Step 5. The last step is when we want to store the string as autonomous DNA record in the EHR. In order to convert $\boldsymbol{c}$ to DNA we map each number {0, 1, 2, 3} to the appropriate letter {A, C, G, T}. As a result we get the sequence

$$\boldsymbol{d} = T\ CGGACGCCCGT\ ACGT\ ACGT\ T\ CCAGCT\ GCCT\ C.$$

The process of encoding data to DNA using Hamming (31, 26) code is finished when all of the above steps are completed.

C. Decoding DNA record to data string using Hamming (31, 26) code

This section will demonstrate all of the steps required for DNA record decoding using the *Hamming (31, 26)* code in order to transmit the DNA record to other EHR systems and successful use it in a secure manner at the final destination.

Step 1. Suppose a DNA strand $\widetilde{\boldsymbol{d}}$ is retrieved after sequencing and that error might have occurred. Therefore, we need to convert $\widetilde{\boldsymbol{d}}$ back to *quads*.

Step 2. Secondly, the *parity-check* matrix $\boldsymbol{H}$ is used in order to calculate an error vector $\boldsymbol{e} = \widetilde{\boldsymbol{b}}\boldsymbol{H}$ when $\tilde{\boldsymbol{c}}$ is converted into $\widetilde{\boldsymbol{b}}$ and the *parity quad* $\widetilde{\boldsymbol{p}}$ and $\widetilde{\boldsymbol{b}}$ are decoded. In this way it is possible to find out the error position and error value.

Step 3. The next step is to fix and decode the data into $\widetilde{\boldsymbol{b}}_{dec}$ which is done by subtracting (*modulo 4*) the error value from the *quad* in order to obtain the original *quad*.

Step 4. The fixed data $\widetilde{\boldsymbol{b}}_{dec}$ will help us to check for *parity quad* $\widetilde{\boldsymbol{p}}$ and based on the previous data the type of the error is determined and returned.

Step 5. By implementing *CRC* we can see if there are any errors left in $\widetilde{\boldsymbol{a}}$. In order to provide this step, $\widetilde{\boldsymbol{a}}$ is split into the first 20 *quads*, denoted by $\widetilde{\boldsymbol{m}}$, and the last 6 *quads*, denoted by $\widetilde{\boldsymbol{h}}$, where *CRC* for $\widetilde{\boldsymbol{m}}$ is computed and compared to $\widetilde{\boldsymbol{h}}$. This check will return *True* or *False*.

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

Step 6. The last step is to convert $\widetilde{\boldsymbol{m}}$ to $\tilde{\boldsymbol{s}}$ in reverse order of *Stage 1* of encoding $\tilde{\boldsymbol{s}}$ until there are no more *quads* to convert. The final string is stored in $\widetilde{\boldsymbol{s}}$ and is returned with the result of the *CRC* check.

The process of decoding DNA record using Hamming (31, 26) code is finished when all of the above steps are completed.

D. Example of error

This section demonstrates how the appearance of one error: non-parity base during the process of transmission to the final destination is handled by the Hamming (31, 26) code. Note that the type of the error in our case is correctable error which is denoted by 2.

$\widetilde{\boldsymbol{d}}$ = T AGGACGCCCGT ACGT ACGT T CCAGCT GCCT C

$\widetilde{\boldsymbol{c}}$ = 3 0220 1211 1230 1230 1233 110213 21131

$\widetilde{\boldsymbol{b}}$ = 0220 1211 1230 1230 1233 110213 21131

err_type = 2

e = [33000]

(*err_pos, err_val*) = (0, 3)

$\widetilde{\boldsymbol{a}}$ = 1220 1211 1230 1230 1233 110213

$\widetilde{\boldsymbol{m}}$ = 1220 1211 1230 1230 1233

CRC pass = *True*

$\widetilde{\boldsymbol{s}}$ = h e l l o

E. Analysis of the encoding and decoding process with Hamming (31, 26) code

The encoding and decoding process of using the *Hamming (31, 26) code* has three ways of securing the data: *Parity check* matrix $\boldsymbol{H}$, *parity quad* that was added and *CRC* pass. Hence, suppose one error occurred during sequencing and $\widetilde{\boldsymbol{d}}$ is retrieved. Using the *parity-check* matrix $\boldsymbol{H}$ it will find and correct that error. However, if the error is detected in *parity quad* $\widetilde{\boldsymbol{p}}$ it will not affect the decoding process and $\boldsymbol{s}$ can still be retrieved.

The second scenario is that two errors occurred in $\widetilde{\boldsymbol{d}}$ and both are not in $\widetilde{\boldsymbol{p}}$. In this case $\boldsymbol{H}$ will detect that there is an error, but if $\boldsymbol{e}$ contains different non-zero values, then multiple errors have occurred, but the code is not able to correct more than one error. Hence, $\boldsymbol{H}$ can only detect and correct one error, so it will incorrectly decode $\widetilde{\boldsymbol{d}}$ **.** Now, the parity quad $\widetilde{\boldsymbol{p}}$ comes into play to check whether the received $\widetilde{\boldsymbol{d}}$ matches with the parity. All calculations are done base on the incorrectly decoded sequence $\widetilde{\boldsymbol{d}}$. If multiple errors occur, that match cannot be not performed, but nevertheless, the CRC usually still detects those errors.

IV. Reed-Solomon

This section uses a concrete example to explain how the Reed-Solomon code encodes and decodes DNA records. Then, it calculates the number of errors detected and corrected during the data transfer process to other EHR systems.

A. Reed-Solomon applied to DNA record

The *Reed-Solomon* codes are used in order to encode the data and the main goal of this technique is to provide correction of multiple errors. In order to apply the *Reed-Solomon* to DNA records, the field *F (256)* is used which requires to choose *n* and *k* such that $\frac{n-k}{2} = 2$. In our case it is logical to choose *n* = 255 and *k* = 251, but the original string $\boldsymbol{s}$ is consisted of 5 symbols which means that it is more convenient to choose *k* = 5 [21]. This is the main reason why a *shortened Reed-Solomon code* is used.

Shortening a *RS (n, k)* code with minimum distance d by a symbols will yield a *RS(n−a, k−a)* code with minimum distance d, where a is a primitive element. Therefore, the *RS (255,251)* code is shortened into a *RS (9,5)* code. Because the encoding and decoding schemes of the *Reed-Solomon* code are fairly complicated we decided to implement the unireedsolomon 1.0 package available on *PyPI* under an *MIT* license that can encode and decode a possible shortened Reed-Solomon code over *F (256)* for a given *n* and *k*.

B. Encoding data string to DNA record using RS (9, 5) code

This section demonstrates all steps required for data string encoding using the *RS (9, 5)* code to save the DNA record and transmit it to the EHR system safely to the final destination.

Step 1. Firstly, the string **s** = *hello* is converted to *ASCII* symbols, and then, using the field *F (256)* the symbols are converted to values between 0 and 255. So, the resulting message is denoted by

$$\boldsymbol{m} = 104\ 101\ 108\ 108\ 111.$$

Note that the length of $\boldsymbol{m}$ is five in this case, the symbols consist of 3 digits.

Step 2. Secondly, the message $\boldsymbol{m}$ is encoded with *RS(9,5)* code with the previously mentioned package using a generator polynomial q which as a result will return a string with 9 symbols given as

$$\boldsymbol{b} = 104\ 101\ 108\ 108\ 111\ 127\ 24\ 174\ 193.$$

Step 3. In this step all of the *ASCII* symbols are converted to *base 4* numbers and they are concatenated and the following sequence is obtained

$$\boldsymbol{c} = 122012111230123012331333012022323001.$$

If we compare the result what we have obtained in *Section III.B* we can conclude that the value of $\boldsymbol{s}$ is changed from 32 to 36 *quads*.

Step 4. The last step is to convert $\boldsymbol{c}$ to autonomous DNA record ready for storing or transferring to other EHRs. The conversion is done by mapping the numbers $\{0,1,2,3\}$ into appropriate letters $\{A, C, G, T\}$, respectively. As a result we get

$\boldsymbol{d}$ = CGGACGCCCGT ACGT ACGT T CT T T ACGAGGT GT AAC.

The process of encoding data string to DNA record using *RS (9, 25)* code is finished when all of the above steps are completed.

C. Decoding DNA record to data string using RS (9, 5) code

This section will demonstrate all of the steps needed for DNA record decoding using the same *RS (9, 5)* code in order to transmit the DNA record to other EHR systems and successfully use it in a secure manner at the final destination.

Step 1. Suppose a DNA strand $\widetilde{\boldsymbol{d}}$ is retrieved after sequencing and that error might have occurred. Therefore, we need to convert $\widetilde{\boldsymbol{d}}$ back to *quads* to gain $\widetilde{\boldsymbol{c}}$.

Step 2. Secondly, $\widetilde{\boldsymbol{c}}$ is divided into parts of 4 *quads* and each of them is read as a *base 4* number and converted to *ASCII* symbol, and after that it is converted to element sequence returned from the field *F (256).* When we put both together we get that $\widetilde{\boldsymbol{b}}$ has length 9.

Step 3. The next step is to fix and decode the data into $\widetilde{\boldsymbol{b}}$ which will give us the value of $\widetilde{\boldsymbol{b}}_{dec}$. This process is done via the *Reed-Solomon* decoding function from the package mentioned above.

In order to check that $\widetilde{\boldsymbol{b}}_{dec}$ is a valid code, or is decoded without any errors, the *RS chec*k is introduced. All codewords are multiples of the generator polynomial *g*, so $\widetilde{\boldsymbol{b}}_{dec}$ is a codeword if *g* divides $\widetilde{\boldsymbol{b}}_{dec}$. This check will return *True* or *False*.

Step 4. The last step is to translate the ASCII symbols back to characters to form string $\widetilde{\boldsymbol{s}}$.

The process of decoding data to DNA using *RS (9, 25)* code is finished when all of the above steps are completed.

D. Example of error

This section demonstrates how the appearance of *two errors in different parts* is handled by the *RS (9, 5)* code.

$\widetilde{\boldsymbol{d}}$ = $\underline{\text{A}}$GGA$\underline{\text{A}}$GCCCGT ACGT ACGT T CT T T ACGAGGT GT AAC

$\widetilde{\boldsymbol{c}}$ = 0220 0211 1230 1230 1233 1333 0120 2232 3001

$\widetilde{\boldsymbol{b}}$ = 40 37 108 108 111 127 24 174 193

$\widetilde{\boldsymbol{b}}_{dec}$ = 104 101 108 108 111 127 24 174 193

RS check = True

$\widetilde{\boldsymbol{m}}$ = 104 101 108 108 111

$\widetilde{\boldsymbol{s}}$ = h e l l o

E. Analysis of the encoding and decoding process with Reed-Solomon (9, 5)

The encoding and decoding process of using the *Reed-Solomon (9,5)* and choosing the parameters *n* and *k* specifically can determine how many errors the code can correct in the field which is applied. However, in our case we need to correct 2 base errors. By applying the scheme to the DNA records, it can correct up to 8 base errors because the Reed-Solomon code works in field *F(256)* which can correct 3 errors in the ASCII symbols. Note that the main goal to implement this type of code is the ability to correct multiple errors if they occur into the data during the transmission process.

V. Comparison

In this section, the results from the previous sections are summarized and compared. When analyzing the error detection and correction, the *Reed-Solomon* code clearly outperforms the *Hamming code* since it can correct 2 errors instead of 1. Therefore, as explained in the previous section, the Reed-Solomon code in many cases can correct even more than 2 errors when they occur in the same DNA records that correspond to one symbol of *F(256).*

The main difference between the encoding schemes of the *Hamming* and *Reed-Solomon codes* is the order in which the data is encoded and is converted to *quads*. For the *Hamming code*, the string **s** was first converted to *quads* before the *CRC*, and the matrix G was used to encode the message. On the other side, for the *Reed-Solomon code*, the string was firstly encoded and converted to *quads* afterwards, which also changed the order of the decoding steps. Table 2 gives the differences and similarities between both coding schemes used in EHR systems.

Hamming Code	***Reed-Solomon Code***
Defined in the binary field	Defined in the non-binary field
Correct one dedicated error	Correct multiple dedicated errors
Unique steps for Encoding/Decoding process (see Section III)	Unique steps for Encoding/Decoding process (see Section IV)
Poor Performances and process small data	Better Performances and process bigger data

Table 2*: Hamming Code vs. Reed Solomon applied in EHR systems*

VI. Conclusion

DNA record is very sensitive patient information. Keeping this data in its original format in the EHR systems and transfer it to other medical centers unchanged for preparing health treatment is an important issue which needs to be addressed. Introducing coding schemes for storing and processing data can guarantee the reliability and security of those systems. Therefore, in this paper, we examine and demonstrate the usefulness of two different code schemes, the Hamming and Reed-Solomon. The methods used in this paper were based on a *Ham (31, 26) code* and a *RS (9, 5) code*. Both schemes were introduced with descriptions for the encoding and decoding steps. Through examples this paper investigated to see how they responded to data errors.

The conclusion is that the *Reed-Solomon* outperform the *Hamming code. The RS code* excels in other important qualities like error correction, and its implementation is not as simple as the implementation of the *Hamming code*. On the other side, the *Reed-Solomon code* has more potential to work properly on more enormous data sets where the number of errors that the code is able to correct can increase. Counting up all these arguments, it is fair to conclude that the *Reed-Solomon code* is more suitable for transmitting the DNA record to the EHR system than the *Hamming code*.

REFERENCES

[1] Mitreska I., Marina N., Capeska Bogatinoska D. (2021) Electronic Health Records System for Efficient Healthcare Services. In: Badnjevic A., Gurbeta Pokvić L. (eds) CMBEBIH 2021. CMBEBIH 2021. IFMBE Proceedings, vol 84. Springer, Cham. https://doi.org/10.1007/978-3-030-73909-6_

[2] A. Blanco and G. Blanco, “Nitrogenous Base”, Science Direct, 2017

[3] C. N. Takahashi, B. H. Nguyen, K. Strauss, and L. Ceze, “Demonstration of End-to-End Automation of DNA Data Storage,” Scientific Reports, vol. 9, pp. 4998, 2019

[4] S. Mahmood, M. J. Khan and S. Anwer, “Applications of Security Reference Architectures in Distributed Systems: Initial Findings of Systematic Mapping Study”, Tenth International Conference on Software Engineering Advances, 2015

[5] J. A. Milstein, J. Everson, S. D. Lee, “EHR Adoption and Hospital Performance: Time-Related Effects”, Health Research and Educational Trust, vol. 50, pp. 1751-1771, 2015

[6] I. Shadmanov and K. Shadmanova, “Summarization of Various Security Aspects and Attacks in Distributed Systems: A Review”, Advances in Computer Science: an International Journal, vol. 5, no. 19, January, 2016

[7] E. Bertino, “Introduction to Data Security and Privacy”, Springer, pp. 125-126, September, 2016

[8] G. F. Elkabbany and M. Rasslan, “Security Issues in Distributed Computing System Models”, IGI Global, 2017

[9] K. E. M. Msiska, A. Kumitawa and B. Kumwenda, “Factors affecting the utilization of electronic medical records system in Malawian central hospitals”, M. M. Journal, pp. 247-253, September, 2017

[10] D. Wani and M. Malhota, “Does the meaningful use of electronic health records improve patient outcomes?”, Journal of Operations Management, vol. 60, pp. 1-18, June, 2018

[11] C. S. Kruse, A. Stein, H. Thomas and H. Kaur, “The use of Electronic Health Records to Support Population Health: A Systematic Review of the Literature”, M. M. Journal, September, 2018

[12] V. C. Hu, D. R. Kuhn and D. F. Ferraiolo, “Access Control for Emerging Distributed Systems”, National Institute of Standards and Technology, 2018

[13] S. Chenthara, K. Ahmed, H. Wang and F. Whittaker, “Security and Privacy-Preserving Challenges of e-Health Solutions in Cloud Computing”, IEEE Access – Multidisciplinary open access journal, May 2019

[14] E. Lupu, “Distributed Systems Security”, The National Cyber Security Centre, October, 2019

[15] N. Paunkoska and A. Risteski, “Security in distributed storage systems – Encryption algorithm vs. Coding schemes”, Journal of Electrical Engineering and Information Technologies, November, 2019

[16] T. R. Schopf, B. Nedrebq, K. O. Hufthammer, I. K. Daphu and H. Laerum, “How well is the electronic health record supporting the clinical tasks of hospital physicians? A survey of physicians at three Norwegian hospitals”, BMC Health Services Research, December, 2019

[17] L. Li, C. Chang, J. Bai, H. Le, C. Chen and T. Meen, “Hamming Code Strategy for Medical Image Sharing”, Applied System Innovation – An Open Access Journal from MDPI, January 2020

[18] M. Blawat, K. Gaedke, I. H¨utter, X.-M. Chen, B. Turczyk, S. Inverso, B. W. Pruitt, and G. M. Church, “Forward Error Correction for DNA Data Storage,” Procedia Computer Science, vol. 80, pp. 1011–1022, 2016

[19] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, “Robust chemical preservation of digital information on DNA in silica with error-correcting codes,” Angewandte Chemie - International Edition, vol. 54, pp. 2552–2555, 2 2015

[20] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen, C. N. Takahashi, S. Newman, H.-Y. Parker, C. Rashtchian, K. Stewart, G. Gupta, R. Carlson, J. Mulligan, D. Carmean, G. Seelig, L. Ceze, and K. Strauss, “Random access in large-scale DNA data storage,” Nature Biotechnology, vol. 36, pp. 242–248, 2 2018

[21] E. Slingerland, “DNA Data Storage Hamming and Reed-Solomon Codes”, Delft, Nederlands, June 2019