

# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900

Open access books available

186,000

International authors and editors

200M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index  
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?  
Contact [book.department@intechopen.com](mailto:book.department@intechopen.com)

Numbers displayed above are based on latest data collected.  
For more information visit [www.intechopen.com](http://www.intechopen.com)



# Information Encoding for Flow Watermarking and Binding Keys to Biometric Data

*Boris Assanovich, Iryna Korlyukova and Andrei Khombak*

## Abstract

Due to the current level of telecommunications development, fifth-generation (5G) communication systems are expected to provide higher data rates, lower latency, and improved scalability. To ensure the security and reliability of data traffic generated from wireless sources, 5G networks must be designed to support security protocols and reliable communication applications. The operations of coding and processing of information during the transmission of both binary and non-binary data in nonstandard communication channels are described. A subclass of linear binary codes is considered, which are both Varshamov-Tenengoloz codes and are used for channels with insertions and deletions of symbols. The use of these codes is compared with Hidden Markov Model (HMM)-based systems for detecting intrusions in networks using flow watermarking, which provide high true positive rate in both cases. The principles of using Bose-Chadhuri-Hocquenghem (BCH) codes, non-binary Reed-Solomon codes, and turbo codes, as well as concatenated code structures to ensure noise immunity when reproducing information in Helper-Data Systems are considered. Examples of biometric systems organization based on the use of these codes, operating on the basis of the Fuzzy Commitment Scheme (FCS) and providing  $FRR < 1\%$  for authentication, are given.

**Keywords:** linear codes, Varshamov-Tenengoloz codes, non-binary turbo codes, Reed-Solomon codes, concatenated codes, flow watermarking, biometric system

## 1. Introduction

Engineers and researchers around the world have been using various error correction codes (ECCs) for almost a century to provide communication and combat noise in information channels. In addition to communication, ECCs have found many other uses, including watermarking and intrusion detection, cryptography, and information security. Digital watermarking is the process of embedding a digital code into some public data. Today, this technology is widely used not only in multimedia processing but also in network traffic monitoring. In this case, the input patterns, which are easily identified when the watermarked flows cross an observation point, allow the creation of a mechanism to scan the network for the harmful activity. This procedure finds applications both for securing network connections and intrusion detection in them.

On the other hand, when providing secure access to any data, it becomes necessary to use user verification by analyzing his password, which requires ensuring the reliability of its storage. To solve this problem, biometric methods of organizing secure access to the system are widely used, that reduce the risks of storing passwords, which have long been a weak point in security systems. This chapter will discuss some types of the ECC and how they can be used to help ensure the security and reliability of information.

In recent years, the technique of applying the ECC has been undergoing changes due to the use of machine learning (ML) methods and, in particular, deep learning (DL). A good review of the recent advancements in DL-based communication was made by Qin et al. [1], where the authors described the use of this technique for channel modeling, modulation recognition, and improvement of decoding methods. In recent papers, the authors have considered in more detail the DL methods for decoding known codes [2] and, moreover, for constructing an ECC based on intelligent methods [3]. Despite the increasing use of the ML technique for ECC, it is important to understand both the principles of describing known ECC based on algebraic constructions that lead to elegant decoding algorithms and their application in non-standard communication channels.

The rest of the chapter is organized as follows. First, we present the basic encoding-decoding principles of the binary and non-binary ECC used for substitution and symbol insertions and deletions errors in Section 2. Then we discuss the flow watermarking techniques for intrusion detection in Section 3. In Section 4, we describe the use of various ECC types in biometric systems (BSs) for solving the problem of authentication and present our conclusion in Section 5.

## 2. Error-correcting codes

### 2.1 Linear codes

At the present stage of the ECC theory and technology development, more and more complex code structures attract our attention. Although coding algorithms are becoming more complex and require powerful computing resources, in recent years, researches have increasingly turned to known codes and mathematical descriptions developed for them. Such codes, for example, are *linear codes*, which have useful properties and can be used in non-standard data transmission channels applications.

There are many good tutorials about error-correcting codes (for example, see [4, 5]), so only the necessary definitions are used in the entire chapter. We define a *code*  $C$  of block length  $n$  over an alphabet  $q$  is a subset of  $q^n$ , together with a one-to-one encoding which maps a message set  $M$  to a code set  $C$ . The main goal of encoding is to increase the resilience of the messages to errors, where  $|C|$  denotes the number of elements in a set or the code cardinality.

We start from the description of linear code. A linear  $q$ -ary code of length  $n$  and dimension  $k$  is a linear subspace  $C$  with dimension  $k$  of the vector space with dimension  $n$ , whose elements are the elements of the field  $GF(q)$ . The description of the properties of linear codes will be done on the example of binary codes, whose symbols are the elements of a field  $GF(2) = \{0;1\}$  which is a code alphabet.

Generally, a binary code  $C$  is defined as a set of finite sequences (vectors)  $\mathbf{x} = (x_1, \dots, x_n)$ , called codewords, encoded with the use of corresponding message vectors  $\mathbf{b} = (b_1, \dots, b_k)$  from code symbols  $x_i, b_i \in GF(2)$ . Linear  $(n, k, d)$ -code is

defined by following parameters: Hamming distance between binary codewords  $d(\mathbf{x}_i; \mathbf{x}_j)$ , weight of a codeword  $wt(\mathbf{x}_i)$  and a code rate or coding efficiency  $k/n$ . Linear codes are defined by their generator and parity-check matrices  $\mathbf{G}$  and  $\mathbf{H}$ , respectively, whose columns and rows are linearly independent. Every codeword is a linear combination of rows of the generator matrix  $\mathbf{G}$ . The *minimum distance*  $d_{min} = \min\{d(\mathbf{x}_i; \mathbf{x}_j)\}$  of a linear ECC and its code weight distribution define its error correction capacity  $t$  or maximum number of symbols that can be corrected in a codeword. There is a simple method of minimum distance decoding with syndrome that could be applied in order to correct  $t$  or less errors in a codeword.

According to this principle, the decoder selects a codeword to minimize the Hamming distance of the matched codeword relative to the received codeword  $\mathbf{y}$  using a reduced look-up table. This is allowed by the linear property of the code.

The decoder performs following steps: the syndrome calculation of codeword  $\mathbf{y}$ :

$$\mathbf{S} = \mathbf{y} \cdot \mathbf{H}^T, \quad (1)$$

determination of the most likely error vector  $\mathbf{e}$ , and estimation of the possibly transmitted codeword  $\mathbf{x}^*$ . Next, the decoder selects that vector  $\mathbf{e}$  of the smallest weight that satisfies  $\mathbf{e} \cdot \mathbf{H}^T = \mathbf{S}$ . These syndrome-based decoding procedures are linear and of low complexity, and only the second step requires a non-linear look-up table operation. In the case of linear codes use, the so-called standard arrays are widely applied, which make it possible to find the corresponding codeword for any received vector.

The standard array for a binary  $(n, k)$  code is an array of size  $2^{n-k}$  by  $2^k$  where: (1) the first row has the codewords with “all zeros” on the left); (2) the 1st column is a coset leader for a coset in each row; and (3) the entry in the  $i$ -th row and the  $j$ -th column is the sum of the  $i$ -th adjacency coset leader and the  $j$ -th codeword. However, the linear property of the code allows the use of syndrome decoding, which is an efficient decoding technique using a reduced look-up table.

For linear codes, it is important that the number of syndromes,  $2^{n-k}$ , must be greater than or equal (for perfect codes) to the number of correctable error patterns  $\sum_{i=0}^t \binom{n}{i} \leq 2^{n-k}$ , which is determined by the so-called Hamming bound [4].

If we take a linear  $(6, 3, 3)$ -code  $C$  with codewords  $\{(000000), (110100), (011010), (101110), (101001), (011101), (110011), \text{ and } (000111)\}$ , obtained on the basis of the generator matrix  $\mathbf{G}$  ([5], pp. 357–367), then there are modification methods to change its properties [5]. For example, the number of its codewords can be increased or decreased. If individual codewords are removed from code set  $C$ , then a new code  $C'$  with the same properties can be constructed while maintaining the minimum weight of codewords. This modified code  $C'$  is a *subcode* of  $C$ .

## 2.2 Cyclic codes

Binary *cyclic codes* are block codes for which cyclic shifts of each codeword yield a different codeword and can be efficiently encoded and decoded using shift registers and combinatorial logic. Cyclic codes are linear codes with good properties and can be defined by polynomials:

$$u(x) = u_0 + u_1x + \dots + u_{n-1}x^{n-1} \quad (2)$$

In such a polynomial representation, the presence or absence of the formal variable  $x$  with a degree is determined by the coefficient and corresponds to the binary “1” or “0” of the codeword element.

Cyclic codes have the property that all code polynomials  $u(x)$  are multiples of a unique polynomial  $g(x)$ , called the *generator polynomial* of the code. This generator polynomial is completely described by its roots, which are called zeros of the code.

Sometimes, to find a generating polynomial  $g(x)$ , the polynomial  $(x^n - 1)$  must be factored into its irreducible factors  $f_i(x)$ . Since a cyclic code is also linear, any set of linearly independent vectors can be selected as a generator matrix. However, in this case, a nonsystematic encoding is performed, when the message bits can appear explicitly in any positions of a codeword. However, the encoding of codewords of a binary cyclic code can be also systematic, if the message is processed in another way. With this encoding, information and check symbols are clearly separated. Another polynomial,  $h(x)$ , called the parity-check polynomial, can be related to the parity-check matrix. Generator polynomial and parity-check polynomial are connected by  $g(x)h(x) = x^n + 1$ .

Then, a parity-check matrix for a cyclic code is given by using as rows the binary vectors associated with the first  $n - k - 1$  nonzero cyclic shifts. In the case of high-rate cyclic  $(n, k)$  codes, say  $k/n > 0.5$ , encoding by the division of  $x^{n-k}u(x)$  by  $g(x)$  or by recursion with  $h(x)$ , the coefficients of  $u(x)$  are in the systematic form so that the first  $k$  coefficients are the message bits and the remaining  $n - k$  coefficients are the control bits. However, for powerful cyclic codes correcting multiple errors, the algebraic decoding procedure becomes much more complicated.

It should be noted that the principles of representation and encoding and decoding of polynomial codes are based on the concepts of both simple and extended finite fields, calculations in which can be found in a number of textbooks [5]. Below, we will only briefly use the basic concepts.

Representatives of more powerful correction codes are the Bose-Chadhuri-Hocquenghem (BCH) codes that provide suitable selection of block lengths, code rates, and correcting capacity. BCH codes are cyclic codes that are constructed by specifying the roots of their generator polynomials, i.e., a BCH code of  $d_{min} \geq 2t_d + 1$  is a cyclic code whose generator polynomial  $g(x)$  has  $2t_d$  consecutive roots  $\alpha^b, \alpha^{b+1}, \alpha^{b+2t_d-1}$ , where  $t_d$  is a designed capacity. Next, the generator polynomial of the BCH  $(n, k, d_{min})$  code is

$$g(x) = LCM\{f_b(x), f_{b+1}(x), \dots, f_{b+2t_d-1}(x)\}. \quad (3)$$

Here, LCM is the least common multiple. Thus, we have a code with a length of  $n = LCM\{n_b, n_{b+1}, \dots, n_{b+2t_d-1}\}$ , and dimension of  $k = n - \deg[g(x)]$  and a designed minimum distance  $2t_d + 1$ , which in the general case can be less than the real minimum distance.

For example, consider  $GF(2^4)$ ,  $p(x) = x^4 + x + 1$ , with  $t_d = 2$  and  $b = 1$ . Then,

$$g(x) = LCM\{(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)\} = x^8 + x^7 + x^6 + x^4 + x + 1. \quad (4)$$

We get a double-error-correcting binary BCH (15,7,5) code.

The main idea of decoding binary BCH codes is to use the elements of  $GF(2^m)$  to number the positions of a codeword that are found by solving a set of equations, which can be obtained from the error polynomial and the zeros of the code. The most



popular methods for decoding BCH codes include the Berlekamp-Massey (BM), Euclid, and Peterson-Gorenstein-Zierler (PGZ) algorithms and are discussed in more detail in [4].

### 2.3 Reed-Solomon codes

Reed-Solomon codes are multiple error-correcting non-binary codes that were introduced by Irving S. Reed and Gustave Solomon in 1960. There are two main representations of Reed-Solomon codes – the original representation and the BCH-based representation, which is the most common, due to the fact that BCH-based decoding is more efficient compared to the original representation decoders. In the first case, if  $u(x) = u_0 + u_1x + \dots + u_{k-1}x^{k-1}$  is given as an information polynomial, and  $u_i \in GF(2^m)$ , then there are  $2^{mk}$  such polynomials obtained after calculating  $u(x)$  over nonzero elements of  $GF(2^m)$ , which are codewords of the  $RS(2^m - 1, k, d)$  code of length  $2^m$ . If we interpret RS codes as non-binary BCH codes and the values of code coefficients are taken from  $GF(2^m)$ , then zeros for a  $t_d$  error-correcting RS code are  $2t_d$  consecutive powers of  $\alpha$ . Moreover, since over  $GF(2^m)$  the minimal polynomials have the form  $f_i(x) = (x - \alpha^i)$ ,  $0 \leq i < 2^m - 1$  and for some integer  $b$ , which usually have values of 0 or 1, we have [4]

$$g(x) = \prod_{j=b}^{b+2t_d-1} (x + \alpha^j), \quad (5)$$

It follows from Eq. (3) that the minimum distance of RS  $(n, k, d)$  code over  $GF(2^m)$  is  $d \geq n - k + 1$ . On the other hand, RS code satisfies singleton bound [4] with equality  $d = n - k + 1$ , which defines it as a maximum distance separable (MDS) code. Since the Reed-Solomon code is a linear code, it is possible to apply the classical coding procedure using its generator matrix.

The decoding algorithms of RS codes are similar to that of binary BCH codes. As shown above, setting the primitive powers of the root as evaluation points makes the Reed-Solomon source code cyclic. Reed-Solomon codes in BCH representation are always cyclic because BCH codes are cyclic. In this regard, they are characterized by the same decoding methods as for cyclic codes. In order to choose the correct algorithm that meets the requirements of the system, it is necessary to understand its purpose, which is determined by the RS decoder operation. There are cycle decoding evaluation algorithm, PGZ algorithm, BM algorithm, Sugiyama algorithm with erasures and without erasures, and list decoding algorithms.

Reed Solomon code can correct not only errors but also the erasures, i.e., so-called “lost” symbols. If  $n_{er}$  symbols of RS code are erased and the remaining  $n - n_{er}$  symbols contain  $n_e$  errors, the BM algorithm can find the correct codeword as long as  $n_{er} + 2n_e \leq 2t < d$ . If  $n_{er} = 0$ , the decoder is used as an errors-only decoder, and if  $0 < n_{er} \leq d - 1$  we can call the decoder as an error-and-erasure decoder (EED) [6].

Sudan in 1997 introduced an algorithm that allows the correction of errors beyond the minimum distance of the code. This algorithm produces a list of codewords (it is a list decoding algorithm) and is based on interpolation and factorization of polynomials over  $GF(2^m)$  and its extensions. The main idea of such decoding is to create a list of possible codewords and apply a list-decoding algorithm with such characteristic as a  $(\rho, L)$ -list, where  $\rho$  is a fractional value of the Hamming distance and  $L$  is the size of the list. It was shown [7] that if the fraction of errors in the received information is at

most  $\rho$ , then the transmitted codeword is guaranteed to be in the output list. Also, note that if  $C$  is  $(\rho, L)$ -list decodable, then we can output at most  $L$  codewords for any received codeword by Sudan algorithm. Application of this algorithm allows to correct  $n - 2\sqrt{nk}$  errors. Several years later, Guswami and Sudan improved the algorithm to correct up to  $n - \sqrt{nk}$  errors [7].

The algebraic decoding methods described above are generally hard decision decoding (HDD) methods, which means that for each symbol a hard decision is made about its value. However, the decoder may also contain an information about the reliability of symbol (for example, the demodulator's confidence in the correctness of the symbol), which allows to build soft decision decoders (SDDs). The advent of turbo codes that use iterated soft decision propagation decoding techniques to achieve error correction efficiency has spurred interest in applying SDD to conventional algebraic codes.

## 2.4 Turbo codes

Turbo codes involve the concatenation of two recursive systematic convolutional (RSC) codes connected serially or in parallel, and an interleaver between them. Due to space limitations in this section, we omit the description of convolutional codes. The iterative decoding of constituent codes starts individually, either serially or in parallel, based on inputs derived from the channel and typically some a priori information. Information from each data symbol propagates through the overall code structure in time. The optimal decoding algorithm for each component code in terms of minimizing the probability of error given independent inputs is the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [8], realizing the maximum a posteriori (MAP) criterion decoding. Then resulting symbol probabilities are used to find the log-likelihood ratio (LLR) for  $q-1$  candidate values when decoding them. Next [8], the most likely element is determined by comparing each LLR value and selecting the symbol with the highest reliability (highest LLR).

## 2.5 VT codes

Often, to describe and compare codes, a channel model is used in which information is transmitted. However, in the presence of noise in the channel, symbols may be received with errors. This type of error sometimes called the substitution error. The influence of interference in communication channels also causes synchronization errors associated with the insertion of additional symbols or deletion of transmitted symbols, which are sometimes called "indels." Therefore, there is a strong reason to develop codes that not only correct substitution errors but also deal with "indels."

One of the first codes to deal with synchronization errors caused by symbol deletion was the Varshamov-Tenengol'ts (VT) codes. Below, we briefly consider this construction.

Given a parameter  $a$ , with  $0 \leq a \leq n$ , the Varshamov-Tenengol'ts (VT) code  $VT_a(n)$  is the set of binary words  $\mathbf{x} = (x_1, \dots, x_n)$  of length  $n$  so that the equality satisfies [9]:

$$\sum_{i=1}^n ix_i \equiv a \pmod{(n+1)}. \quad (6)$$

These codes are single-error-correcting codes and optimal for  $a = 0$  as it was conjectured in [10] and will be discussed below.

For example, after calculation  $\sum_{i=1}^n ix_i \equiv 0 \pmod{7}$  with length  $n = 6$ , we can get VT code set  $VT_0(6) = \{(000000), (001100), (010010), (011110), (100001), (101101), (110011), (110100), (111111)\}$ . Any code  $VT_0(n)$  can be used to communicate reliably over a channel that introduces at most one “indel” in a block of length  $n$ . Levenshtein proposed a simple decoding algorithm [11] based on the deficiency in checksum and weight calculation for a VT code. As an example, assume the code  $VT_0(6)$  is used and  $\mathbf{x} = (110100) \in VT_0(6)$  is transmitted over the channel. If the first bit in  $\mathbf{x}$  is deleted and  $\mathbf{y} = (10100)$  is received, then the new checksum is 4, and the deficiency  $D = 7 - 4 = 3 > wt(\mathbf{y}) = 2$ . The decoder must insert a binary “1” after  $n - D = 3$  “0’s” from the right to get a codeword  $(110100)$ . Such an algorithm for decoding  $VT_0(n)$  code with deletion correction is based on a shift operation and has low complexity.

Considering the simplicity of calculating the parameters of VT codes, we would like to make a linear encoder for efficient mapping of binary message sequences into codewords. For binary VT codes, such an encoder was proposed by Abdel-Ghaffar and Ferriera [12]. They constructed a systematic encoder that maps  $k$ -bit message sequences onto codewords in  $VT_a(n)$ , for  $k = n - \lceil \log_2(n + 1) \rceil$ , where in parentheses is rounding up to a higher integer. In addition, for these codes, the concept of a *syndrome* can be used, which is found as  $Syn(C) \equiv \sum_{i=1}^n ix_i \pmod{n + 1}$ .

Now we can introduce the “parity” bits denoted by  $t_p = n - k = \lceil \log_2(n + 1) \rceil$  and use then in dyadic positions to ensure that  $Syn(C) = a$ . Therefore, the message bits can be encoded by calculating the value of the difference between the desired syndrome and calculated one  $d_C = a - Syn(C) \pmod{n + 1}$ .

In an example, see [12] of code for  $n = 10$  and  $a = 0$ , the parity check and information positions can be represented, respectively, as  $\{1, 2, 4, 8\}$  and  $\{3, 5, 6, 7, 9, 10\}$ , and used to encode  $\mathbf{b} = (011001)$  as follows:  $\mathbf{x} = (x_1x_20x_4110x_801)$ , where.

$x_1 + 2x_2 + 4x_4 + 8x_8 = 0 - (3 \cdot 0 + 5 \cdot 1 + 6 \cdot 1 + 7 \cdot 0 + 9 \cdot 0 + 10 \cdot 1 = 1 \pmod{11})$ . The parity-check sequence of least lexicographic order  $(x_1x_2x_4x_8) = 011000$  can be taken.

However,  $VT_0(n)$  codes are nonlinear, and the dimension of  $k$  for obtaining linear  $(n, k)$  codes is limited as  $k \leq \lfloor n/2 \rfloor$  [13]. Below, we propose an algorithm for finding a linear *substitution and deletion/insertion correction code* from any existed  $VT_0(n)$ . The proposed algorithm is executed step by step as follows: 1) sort the codewords of the code  $VT_0(n)$  in lexicographic order; 2) find and choose  $k$  linearly independent codewords of maximum weight while maintaining  $d(\mathbf{x}_i; \mathbf{x}_j) \geq d_{min}$ ; and 3) construct matrices  $\mathbf{G}$  and  $\mathbf{H}$  from  $C$ , making linear combinations of the selected VT codewords.

Using this algorithm will allow constructing a subcode that has at least  $k + 1$  codewords of the  $VT_0(n)$  code. Obviously, the linear combination of any codeword with itself forms a codeword  $(0 \dots 0)$ , which is also belongs to the code  $VT_0(n)$ . By exploiting the algorithm proposed above, the following generator and parity-check matrixes for the modified  $(6, 3, 3)$ -code  $C'$  have been constructed:

$$\mathbf{G}' = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \mathbf{H}' = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}. \quad (7)$$

Representing  $\mathbf{G}'$  and  $\mathbf{H}'$  as Eq. (5) results in a code set with an increased number of codewords belonging to  $VT_0(6)$  compared to initial code  $C$ . If we discard the elements that are not  $VT_0(6)$  codewords and the codeword  $(000000)$ , and then we get a



subcode consisting of four codewords with desired properties:  $\{(110100), (011110), (101101), (110011)\}$ .

Thus,  $C^*$  is a linear subcode with  $d_{min} = 3$ , at the same time it is a  $VT_0(6)$  code. Therefore, it can be used to correct one substitution error and one “indel” error. At the same time, the analysis showed that its code rate is reduced by about  $\frac{1}{2}$  compared to the code rate of  $C$ . The proposed algorithm [14] can be applied to an arbitrary code to find a correcting VT code, which is a subcode of a linear code. If we take a linear ECC  $(8,2,5)$  [5, p.378], consisting of four codewords, we can also find a linear subcode for it, which is also the code  $VT_0(8)$ . Its properties of one “indel” error and two substitution errors correction are preserved. It is known that the size of any  $VT_0(n)$  is about  $2^n/n$  [6], then additional properties appear, decreasing its rate to less than  $\frac{1}{2}$ .

Recently, these codes have again attracted interest, as evidenced by the publication [15], where an encoding method was proposed for a non-binary systematic VT code.

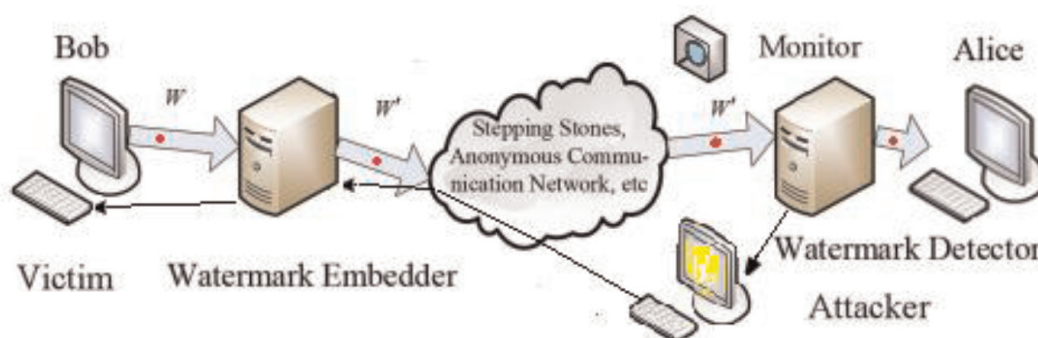
### 3. Use of error-correcting coding in flow watermarking

#### 3.1 HMM-based model for watermark embedding and extraction

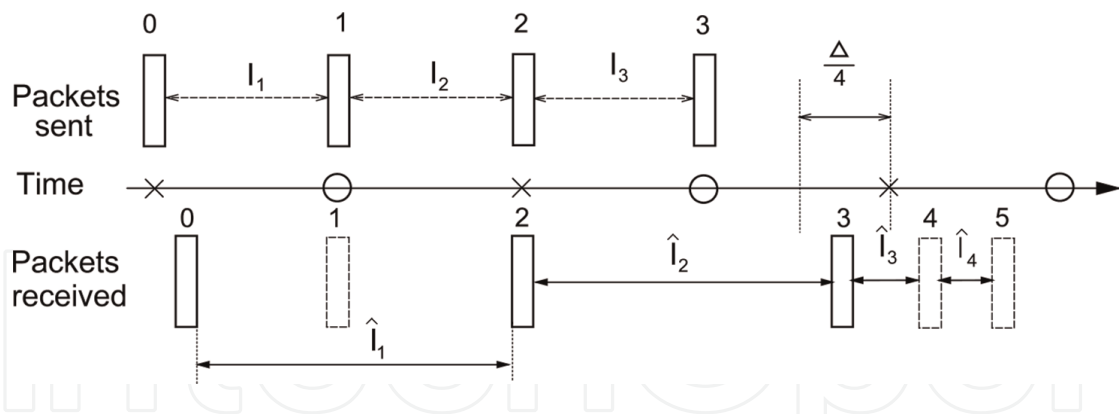
The watermark embedding algorithm aims to detect any changes in the marked data or its integrity. The contents integrity is performed in the verification process. In this section, we discuss the application of ECC for watermark embedding in the context of traffic analysis (TA) used for such purposes as diagnostic monitoring, resource management, and intrusion detection. Intrusion detection systems attempts to detect intrusion through analyzing the network traffic with the use of watermark tracing [16]. If the embedded watermark is both reliable and unique, it is possible to analyze the watermarked return traffic and trace it back at intermediate nodes. This TA approach is referred to as the “flow watermarking” (FW).

To prevent an attacker to endure and analyze the delayed packets and then to eliminate the embedded watermarks, the developed FW schemes have to be “invisible” in the network. An example of stepping-stone detection scenario with FW is depicted in **Figure 1** where an Attacker attacks Victim hiding his identity. Fortunately, FW can be applied for tracing back the attack source.

FW is often implemented on the basis of *inter-packet-delay* (IPD) schemes [17], where watermark bits are embedded in the intermediate packet time which allows to hide traffic artifacts from an attacker. However, in this case, the replacement of



**Figure 1.**  
Attacker detection scenario.



**Figure 2.**  
 An example of IPDs distortion.

packets and packet loss can cause severe detection and decoding errors. The use of ECC makes it possible to improve the noise immunity of FW systems.

The presence of contiguous packet merging leads to a telecommunication channel with deletion and/or substitution errors, and the appearance of jitter-induced bursting or splitting of packets also causes symbol insertions, which requires the appropriate choice of coding for reliable transmission of watermarks. **Figure 2** demonstrates these phenomena. It follows from it that four packets 0, 1, 2, and 3 are sent, three packets 0, 2, and 3 are received, packet 1 is lost, and new packets 4 and 5 are added.

Most FW technologies use a carrier that modulates the transfer of watermark data. Gong et al. [18] embedded quantization index modulation (QIM) watermarks into IPDs and added a layer of ECC to handle watermark desynchronization and substitution errors. Authors developed a Hidden Markov Model (HMM) for channel with dependent deletion and substitution errors using a maximum likelihood decoding (MLD) algorithm paired with a forward-backward algorithm for the calculation of the posterior probabilities [5]. The schematic of the proposed system can be depicted as shown in **Figure 3**.

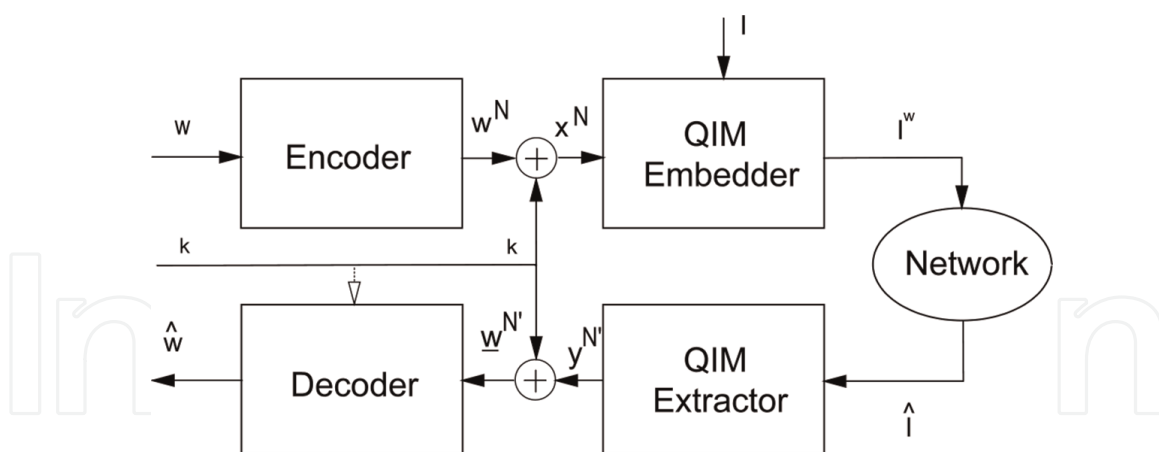
This scheme uses an Encoder and Decoder to process the incoming watermark sequences in order to obtain the codewords  $\mathbf{x}$  of the applied ECC. With this notification, it is implied that the length of a watermark  $w$  and a codeword  $\mathbf{x}$  is defined by user and corresponds to some selected value. Further in the text, superscripts are used to indicate the length of the sequence, and subscripts to determine the element number in a sequence.

Due to the network artifacts described above, the additional transformations must be performed in the encoder to improve noise immunity. For example, see [19], a *sparsification* procedure based on one-hot coding is implemented so that a sparse version of  $w$  is formed, denoted as  $w^N$ , where  $N = sn$  and  $s$  is the *sparsification factor* and has an integer value. This procedure for a channel with the presence of insertions, deletions, and substitutions (IDS) can also be extended for the non-binary case [20].

The  $s$  value is controlled by the density  $f$  that is the ratio of “ones” in  $w^N$ :

$$f = \sum_{i=1}^N w_i^N / N. \quad (8)$$

The whole scheme uses a secret key  $k$ , known to the Encoder and Decoder, which is added bit by bit to  $w^N$  forming a binary sequence  $x^N$  containing one or more codewords.



**Figure 3.**  
System diagram.

The key used for security plays a supporting role in dealing with IDS channel errors during decoding. For example, see [18], if  $w_1 = '1'$ ,  $c = 8$ , then  $x^N = 10000000$  and if  $k = 1111011$  then  $x^8 = 01111011$ . When one bit is dropped and  $y^7 = 0111011$  is received, it can be supposed with a high probability that bit “1” from the second series of bits was lost.

Next, we will consider in more detail the principle of QIM for which modulation and demodulation are carried out by means of QIM Embedder and QIM Extractor, respectively (see **Figure 3**).

To embed the watermark, the IPD flow is modified so that each IPD is converted to an interval according to the even/odd multiplier of the quantization interval  $\Delta/2$ , depending on the value of the 0/1 bit. Formally, this can be represented as:

$$I_i^w = \begin{cases} c\Delta, & \text{if } x_i = 0 \\ (c + 0.5)\Delta, & \text{if } x_i = 1 \end{cases} \quad (9)$$

Since packets can only be delayed by the QIM Embedder, it is possible to define the  $c$  parameter to be the smallest integer so that the change in  $I_i^w$  would slightly delay the  $i$ -th packet. After passing the  $I^w$  sequence through the network, it is received as an estimated IPD sequence  $\hat{I}$  and then analyzed by the QIM extractor, obtaining the necessary information from it.

The following QIM demodulation threshold function is used to recover the embedded bit  $y_i$ :

$$y_i = \begin{cases} \text{mod}(\lfloor 2\hat{I}_i/\Delta \rfloor, 2) & \text{if } 2\hat{I}_i/\Delta - \lfloor 2\hat{I}_i/\Delta \rfloor \leq 0.5 \\ \text{mod}(\lceil 2\hat{I}_i/\Delta \rceil, 2) & \text{if } 2\hat{I}_i/\Delta - \lfloor 2\hat{I}_i/\Delta \rfloor > 0.5 \end{cases} \quad (10)$$

Consider the example in **Figure 2**. Here, the first two IPDs  $I_1$  and  $I_2$  are converted into  $\hat{I}_1$ , and the size of the last IPD  $I_2$  is changed and is determined as  $\hat{I}_2$ . Therefore, the result of the noise in the channel is the bit received before Packet 2, which is due to the two intervals merging  $y_1 = x_1 \oplus x_2$ , and the bit inversion after receiving Packet 3, resulting in  $y_2 = x_3$ .

In general  $y_i = \sum_{j=r+1}^i x_j$  and can take the binary values “0” or “1”, where  $r$  is the index of the last successfully received packet before the  $i$ -th one. As can be seen from

**Figure 2**, two intervals  $\hat{I}_3, \hat{I}_4$  appear resulting in the insertion of new bits into the received watermarked data.

Obviously, in the absence of packet loss or split, the watermark bit is inverted if the IPD jitter exceeds  $\Delta/4$ . The jitter can be described by i.i.d. Laplace distributed with zero mean. Then the jitter substitution error probability can be estimated as:

$$P_s = 1 - F(\Delta/4) = 0.5 \exp\left(-\Delta/2\sqrt{2}\sigma\right), \quad (11)$$

where  $F()$  is the Laplacian pdf and  $\sigma^2$  is its variance. Since packet losses leads to merging of successive IPD, the resulting error contains both deletion and substitution error [18]. In this model, we assume that packets are lost independently and that the initial packet is always synchronized.

Authors in [17, 18] used the concept of *drift* to define the loss of bit synchronization, which is the shift in position of some sent packet in the received flow. Using sparse key parameters, one can determine the probabilities of IDS events in the resulting sequence. These events were interpreted with the use of HMM and applying the forward-backward algorithm [19], the watermark estimation posterior probabilities for the maximum likelihood decoding (MLD) have been derived as.

$$\hat{w}_j = \arg \max P(y^N | w_j), \quad w_j \in \{0, 1\}. \quad (12)$$

After calculating these probabilities for all bits of the watermark sequence, the presence of a watermark in flow is determined based on the correlation value of the resulting sequence and the original one. For those interested in the details of mathematical calculations, one can refer to the original publications of the authors mentioned above.

### 3.2 Use of VT codes in FW

An alternative IPD-FW scheme for embedding watermarks based on the use of binary VT codes, which are subcodes of linear codes and exploiting QIM, has been proposed in [14]. The scheme uses linear codes of length 6 and 8 bits with an attached marker and optional matrix interleaving to deal with bursting errors.

*Coding-decoding scheme without interleaving.* As before, we assume that the watermark  $w$  to be embedded is a bit sequence. Next, the sequence  $w$  is divided into blocks of bits  $\mathbf{b} = (b_1 \dots b_l)$  of length  $l$  and encoded by the chosen VT code  $\mathbf{x}$  of length  $n$  (see above). Then obtained codeword  $\mathbf{x}$  is concatenated with predefined marker pattern  $z$  of length  $m$  making  $w^N$ , where  $N = n + m$ . In this implementation, a pattern  $z$  contains a series of zeros, which is determined by the necessity of XOR-ing all bits of the formed sequence  $w^N$  with a secret key  $k$ , by analogy with the previous HMM-based method. The key  $k$  used is a sparse sequence containing a binary “1” in only one position out of all  $N$  bits. In fact, the sequence  $w^N$  can be made up by the concatenation of  $M$  codewords  $\mathbf{x}$  with a marker  $z$  attached. We denote this composite sequence as  $w^N = w_1 w_2 \dots w_M$ , with which the composite secret key is XOR-ed, forming the sequence  $x^N = x_1 x_2 \dots x_M$ .

Next, the generated sequence  $x^N$  enters the QIM Embedder, where modulation is performed in the same way as described above. Then the IPD sequence  $I^w$  with injected watermark pattern is transmitted and after traversing the network is received



in the form of estimated sequence  $\hat{I}$  and demodulated. The result sequence  $y^N$  is xored with a key sequence  $k$  and serves as an input to the Decoder.

The Decoder detects markers in the  $w^N$  sequence and separates it into codewords of the applied VT code. As a result, the codewords derived from  $w^N$  can contain substitutions, insertions, and deletions. The sequence at the Encoder output  $x^N$  as well as the one that enters the Decoder  $y^N$  in general case do not match. In addition, their lengths may differ, which makes the decoding process difficult.

To solve the problem, it is proposed to use hybrid decoding with error correction and the choice of one of two algorithms is depended on the number of errors in each received codeword  $\mathbf{y}$  [14]. The decoder-type selection is based on an estimate of the codeword  $\mathbf{y}$  length. If the only one “indel” is found, the Levenshtein’s decoding algorithm [11] is used, and if the number of “indel” errors is greater than 1, the MLD  $\mathbf{x}^* = \arg \max \Pr(\mathbf{x}/\mathbf{y})$  is applied. The syndrome decoding (see Eq. (1)) is performed in case of the absence of “indel” errors or after they have been corrected.

For example, suppose that a sequence  $w^N = 110100000.11110000$  at the output of the QIM Extractor processed with the key  $k = 000000000.001000000$  to be decoded using the subcode mapping  $C' = \{(110100), (110011), (011110), (101101)\}$  into message blocks  $\mathbf{b} = \{00, 01, 10, 11\}$ . After detecting a marker and removing it, two codewords  $\mathbf{y}_1 = 110100$ ,  $\mathbf{y}_2 = 11010$  are obtained. The syndrome calculation (Eq. (1)) of  $S = 0$  can serve as a flag that the boundaries of the received word  $\mathbf{y}_1$  are not changed, there are no errors in it, or the number of errors exceeds its corrective capacity. Therefore, it is possible to apply the Levenshtein decoding algorithm to correct the deleted bit in the last position of  $\mathbf{y}_2$ . However, if one more bit is also deleted, after estimating the length  $\mathbf{y}_2$ , it is necessary to proceed to use MLD decoding.

*Coding-decoding scheme with interleaving.* Considering the channel with bursts of errors, the effective mechanisms for separating error bursts are the use of interleaving. We consider an approach using matrix interleaving of a linear subcode of the VT code, which simplifies the decoding process.

It was found in [21] that there is a VT code that coincides with a linear (8,2,5) error-correcting code [5], consisting of four codewords and subcoding  $VT_0(8)$ . However, to perform the independent decoding of codewords from a linear VT subcode, placed in a continuous bitstream, the boundaries of the codewords must be known. We can implement their independent decoding by the organization of the codewords set of linear subcode and the use of matrix interleaving.

The proposed scheme consists of several layers. However, to simplify its work, we describe it based on the scheme in **Figure 3**. As before, we assume that a watermark sequence  $w^N$  is divided into segments of messages  $b = b_1 \dots b_l$  and encoded by the VT Encoder forming codewords  $\mathbf{x}$  of length  $n$ . These codewords are written row by row into an  $Q \times n$  interleaving matrix. Next, each column is concatenated with a predefined marker pattern, which increases the number of matrix rows. Then matrix columns are XOR-ed with the fragments of the secret key  $k$  represented as a sparse binary sequence with a small number of binary ones. The resulting version of  $w^N$  is then read column by column from the interleaving matrix forming a sequence  $x^N$ .

In fact,  $x^N$  is a supercode containing  $Q$  codewords, which is embedded in flow IPD via QIM Embedder. Note that the elements of interleaving and deinterleaving are not shown in the scheme of **Figure 3**. Further, after processing in QIM Embedder, passing through the network, the IPD flow is demodulated in the QIM Extractor and undergoes inverse transformations with respect to encoding (XORing with key, marker removal, finding codeword boundaries, deinterleaving, and decoding). For

$P_d$ (synthetic traffic)	1%	2%	3%	10%	20%
HMM-based	1.000	1.000	1.000	0.994	—
VT code	0.999	0.999	0.999	0.995	0.666

**Table 1.**  
TPR values for varying  $P_d$  with  $FPR < 1\%$ .

information, various interleaving schemes and adjacent deletions correction constructions have been discussed in [22].

Two FW methods have been modeled: the first one is based on HMM and the second one uses VT codes with markers. At the same time, in the first method, the length of the sparse sequence for FW was 10 bits, and in the second method, it was 9 bits, considering the  $VT_0(6)$  code with 3-bit marker. About 5000 packets were generated, in which network jitters were modeled as Laplace distribution with zero mean and a standard deviation of 10 ms. In the synthetic channel, substitution errors followed sequentially after deletion errors, and symbol insertions were studied separately. The detection threshold was chosen to keep false positive rate (FPR) below 1% for all deletion probabilities. The evaluation of true positive rates (TPRs) in the detection of watermarks for two schemes with respect to different deletion probabilities  $P_d$  is presented in **Table 1**.

It follows from **Table 1** that the use of less complex VT coding leads to virtually the same performance compared to HMM. From the results, the TPR value drops to 66% when the packet loss is 20%, which is rare in a network environment. Methods using interleaving and code (8,2,5) showed better results [21] for channels with bursting insertion errors.

#### 4. Application of error-correcting codes in biometrics

In recent years, there has been increasing interest in cryptographic approaches using biometric measurements. For these purposes, many physical methods are used: from taking fingerprints of a person to the dynamics of his gait. The uniqueness of these characteristics allows them to be used for both identification and authentication. However, for the verification organization, it is required to perform the recognition procedure. A special *biometric template*, which is a mathematical representation of features from the original data, is used to store biometric characteristics.

In this section, we will focus on the processing of biometric features of a person's face. Face recognition is very flexible and can be performed from a distance. These systems can be classified as follows [23]: image-based matching (whole face), feature-based face recognition, and video-based matching. The accuracy of the user's biometric data recognition is high. However, the security and privacy of user data may be compromised. In this case, the concept of *cancelable biometrics* is applicable.

The idea of a reversible template was proposed by Ratha et al. [24]. It includes five main features: tautology, irreversibility, accuracy, diversity, and revocability.

There are several approaches to the creation of biometric system (BS), which are based on direct generation of a secret key from biometrics or key binding to biometric

data. The widespread implementation of BS solutions is constrained by the fuzziness of biometric data. This problem can be alleviated by applying error correction codes. Below we will consider several BS based on the use of different methods for obtaining biometric features and various code structures using the so-called *Fuzzy Commitment Scheme* (FCS) [25].

#### 4.1 Biometric system based on facial HOG features

The use of ECC is due to the spread of biometric measurement values, which can be regarded as noise added to the received signal. Taking into account the signal processing procedures for registration and verification, the generalized scheme can be represented as shown in **Figure 4**.

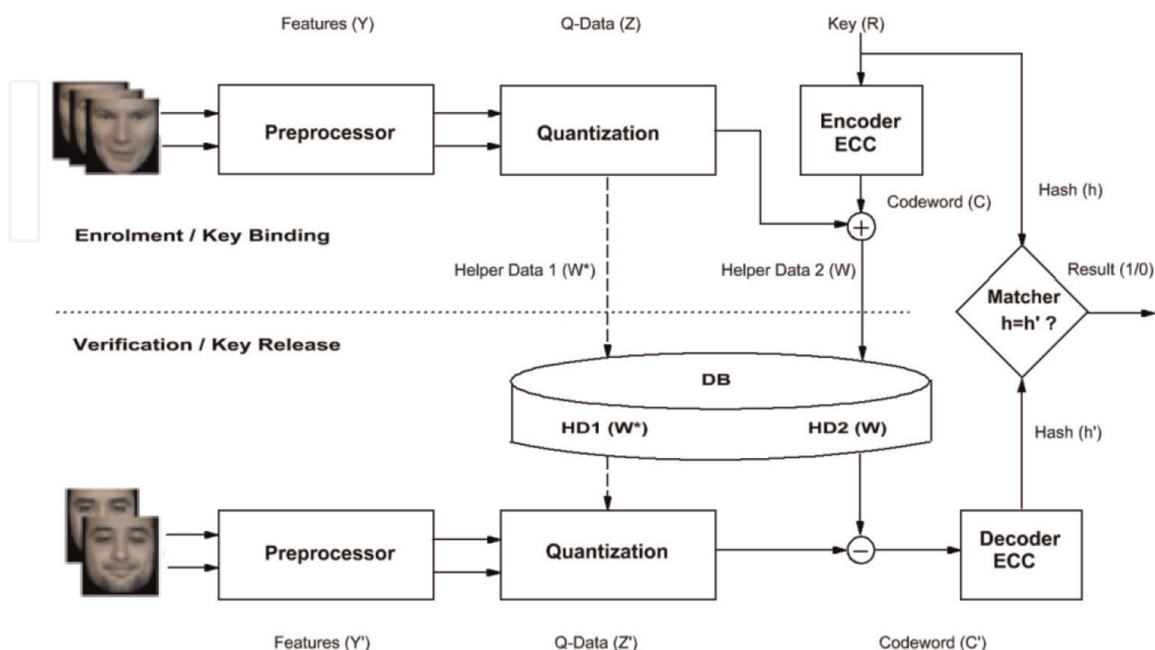
Let us consider the operation of the BS in accordance with [26] with the only difference that instead of local information from convolutions with Gabor kernels, the histogram of oriented gradients (HOG) is used as features. In addition, more powerful BCH codes are used to suppress noise due to fuzzy biometric data [27].

The principle of the scheme operation is as follows. The Preprocessor receives the set of images of the user's face as input, scales them, and converts color images into gray scale ones. Next, HOG features are extracted from the images in the form of real  $Y$  sequences, which can be represented as vectors with a dimension of 4464 elements.

The Preprocessor calculates mean  $\vec{\mu}_i$  and variance  $\vec{s}_i$  for the series of biometric data samples submitted by each  $i$ -th user, as well as the global mean  $\vec{\mu}$  for all registered users.

In addition, the reliability function  $R_i$  is calculated here for each bit  $p$  of each user in according to the expression:

$$R_{i,p} = \frac{1}{2} \left( 1 + \operatorname{erf} \left( \left\{ \left( \vec{\mu}_i \right)_p - \left( \vec{\mu} \right)_p \right\} / \sqrt{2s_{i,p}^2} \right) \right). \quad (13)$$



**Figure 4.**  
Diagram of a biometric system.

Based on the calculated parameters (Eq. (13)), according to  $Y_j$  values, a *mask* is formed containing reliable numbers of data positions in  $Y$  based on a selected threshold. As a result, the obtained values of  $\vec{\mu}_i$ ,  $\vec{\mu}$ , and mask information form Helper data 1 ( $W^*$ ) that are written to the database (DB) before the Quantization procedure. The Quantizer performs data binarization according to Eq. (14) forming  $Z$  sequence of length  $n$ :

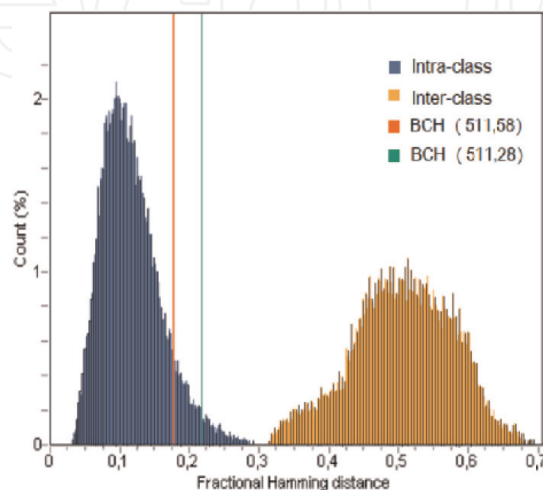
$$(Q_i)_p = 0, \text{ if } (\vec{\mu}_i)_p \leq (\vec{\mu})_p, \text{ and } (Q_i)_p = 1, \text{ if } (\vec{\mu})_p > (\vec{\mu})_p \quad (14)$$

At the same time, a codeword  $C$  and its hash value  $h = h(C)$  are formed by using the ECC Encoder from the user secret key  $R$ . Then a Helper data 2 ( $W$ ) is calculated using the XOR operation as follows  $W_i = C_i \oplus Z_i$ . As a result, the enrollment procedure is completed and the values  $W^*$ ,  $W$ ,  $h(C)$  are entered into the database.

When implementing the user verification, one or more images are sent to the Preprocessor, where they are converted into a sequence of real numbers  $Y'$ . Based on  $W^*$ , reliable positions are determined,  $Q_j$  data values are binarized, and  $Z'$  is obtained. Next, the values of HD2 are retrieved from the database, and operations  $C'_i = W_i \oplus Z'_i$  are performed. The codeword  $C'$  is decoded by the ECC Decoder, and its hash value  $h' = h(C')$  is calculated. Verification is considered successful if  $h(C)$  and  $h(C')$  matches and the corresponding user key  $R$  is extracted.

The OpenFace tool [28] was used to obtain the HOG characteristics of user images containing  $12 \times 12$  blocks of 31 histograms and written into a row vector  $Y$  of length 4464 real values. BCH codes (511,58) and (511,28) over  $GF(2^m)$  were applied as ECC, correcting  $t_d = 91$  and  $t_d = 118$  errors. For performance testing, the Caltech database was used with face images of 24 users. Inter-class and intra-class distributions of the fractional Hamming distance were obtained, which, together with the verticals of the applied BCH codes, are shown in Figure 5.

The calculated values of false acceptance rate (FAR) and false rejection rate (FRR) had the following values: FRR = 0, FAR = 3.5% demonstrating good performance of the used BCH codes, allowing to choose the lengths of secret keys  $K_1 = 58$  and  $K_2 = 28$  bits. Obviously, the length of the  $K_2$  key is too small to register a large number of users. It is clear that after binary quantization the real data are highly rounded, which leads to significant quantization noise. To adapt to biometric real features, we further used unquantized real data processing and non-binary turbo encoding.



**Figure 5.**  
 Inter-class and intra-class distributions.



## 4.2 Application of turbo codes in biometric systems

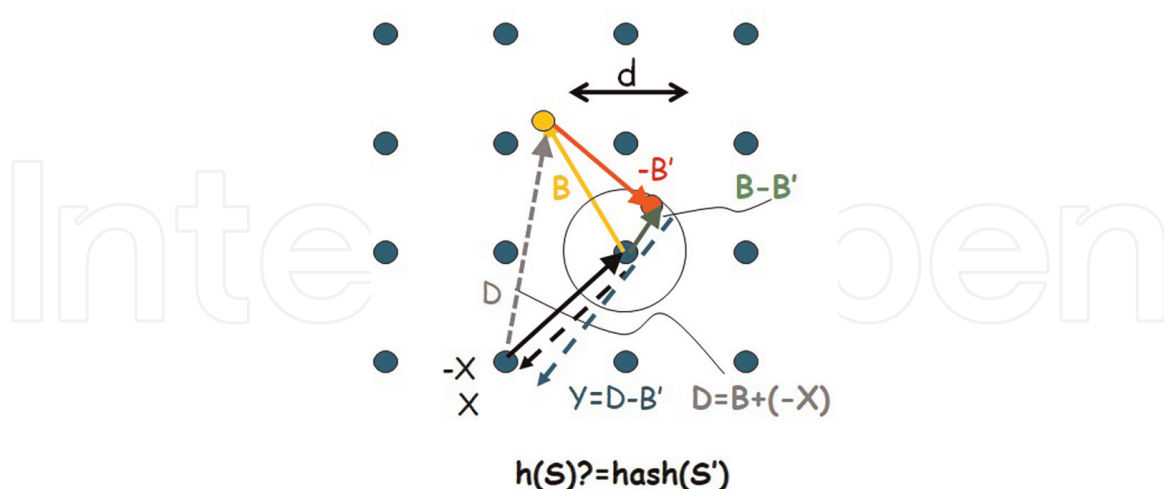
Recently, the use of non-binary turbo codes with modulation in a biometric system has been studied [29]. These codes were generated from non-binary convolutional component codes combined with a random interleaver. Then phase-shift keying (8-PSK) was used. During processing, the polynomials with a coding matrix  $g = [166;176]$  over the ring  $GF(8)$  were used, which made it possible to implement a systematic turbo code with rate of  $1/3$ . As a result, a random octal secret key of length 166 was encoded in turbo code with trailing zeros, forming the resulting  $3 \times 172$  matrix at the output, that was then modulated into the 8-PSK constellation [30]. Each symbol of turbo code  $X$  was presented by I-Q complex numbers giving framed data matrix  $3 \times 344$ . To get biometrical face features, the Caltech database has been used. Data from 511 real numbers obtained after masking procedure (see above) to get components of 4464-element HOG vectors have been used as biometric raw data  $B$ . Then the quantized data with the interval  $q = 0.19635$  was normalized and linearly mapped to the interval  $[0, 2\pi)$  of angles presented then by 2 I-Q components. Hence, the hashed value of result code block together with quantized real data is put into public DB.

At the authentication stage, the resulted codeword  $Y$  corrupted by “biometric noise”  $B'$  is iteratively decoded by the modified BCJR algorithm giving the user password and a hash value. The main operations on 8-ary data blocks (vectors) according to the principles of the BS scheme are shown in **Figure 6**.

Preliminary experimental estimates of FRR resulted in value  $FRR \sim 0.1\%$ , which is several times better than the previous scheme and known results for turbo codes [31].

A further increase in the effectiveness of BS is possible by increasing the inter-class differences in biometric characteristics, which prompted the use of neural networks (NNs) in this area.

In the NN-based system below, we have applied the stacked autoencoder (SAE) structure and the concatenated ECC using RS codes.



**Figure 6.**  
*Vector processing of modulated real data for turbo codes.*

## 4.3 Smiling face biometric authentication system

In the following BS [6], we consider the use of a stacked autoencoder (SAE) to extract features from a sequence of video frames of a user smiling face in order to authenticate him and provide the access to digital services. In contrast to the generally

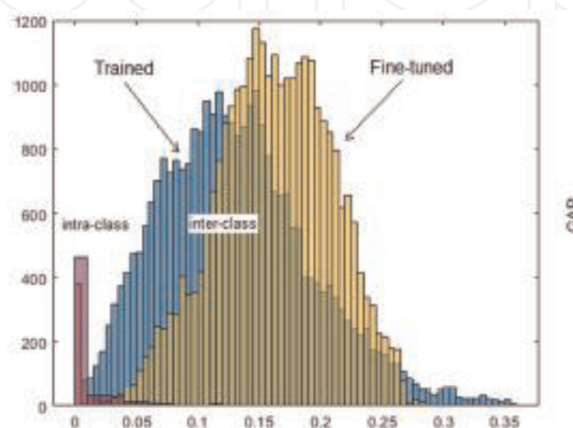
accepted application of binary ECC in BS, we used the non-binary Reed-Solomon codes concatenated with the binary linear ones. The use of these codes, taking into account the dimension of the symbols, leads to an increase in the length of their bit representation. On the other hand, in order to neutralize “biometric noise” and correct errors, it is necessary to increase the ECC redundancy, which reduces the encoding efficiency.

The biometric templates are created according to FCS [25] based on the equidistant quantization of real data at the output of SAE for further processing and encoding by concatenated RS codes. The BS uses concatenated ECC based on non-binary RS codes and binary linear codes with the use of hard decision decoding (HDD) technique and soft decision decoding (SDD) obtained from symbol reliability. The operation principle the proposed BS will be described based on the generalized authentication scheme shown in **Figure 4**.

The Preprocessor block performs such operations as video data capturing, face and smile detection, the smile frames extraction, image transformation and normalization, and features extraction using the SAE pretrained at the registration stage. The biometric data samples obtained from the SAE output layer form the concatenated supervector  $Y = \{Y^1, \dots, Y^M\}$  from several vectors  $Y^i$ , where  $M$  is a number of processed frames. At the stage of Quantization, the real data  $Y$  are converted into their quantized versions  $Z$  producing also deviations  $W^*$  of data values relative to their mean values or centers of quantization intervals used as HD1. In the Encoder block, the user’s password or Key  $R$  is encoded with one or more ECC codewords, depending on the required password strength and FRR. Further, for a biometric authentication purposes, the bit representation of the resulting codeword is XOR-ed with the encoded version  $C$  of quantized data  $Z$ , which results in  $W$  that serves as HD2.

At the Enrollment stage, the biometric samples obtained at the SAE output using HD1 and HD2 then are binded to the secret Key  $R$ . In addition, the  $h$ -hash value of the codewords is calculated and stored in the biometric database DB. During verification, the reverse process of decoupling the “auxiliary” data HD1, HD2, decoding codewords  $C'$ ,  $h'$ -hash calculation, and comparison of two hashes  $h$  and  $h'$  are performed.

A series of experiments were carried out with SAE to get good compact biometric features. To reduce time spent, in these experiments, the subsets of 40 subjects were randomly selected from the entire UvA-NEMO database [32], reproducing a user smile. Unsupervised learning results and then supervised tuning of SAE with parameters 127/63 in the form of histograms are shown in **Figure 7** showing the significant expansion of the inter-class distributions relative to each other.



**Figure 7.**  
 Shift of inter-class distributions during SAE training.

The expected values of FAR and FRR were estimated based on the block error probability of decoding for the uncorrectable error patterns accepted by BS. In the evaluation, we conducted simulation experiments for different error-correcting code structures. The results are placed in **Table 2** [6].

From **Table 2**, it follows that reducing the RS code length makes it possible to increase the performance of the BS in terms of the FRR parameter. Simulation experiments have shown the possibility of achieving the FRR less than 1% for key lengths of 90–170 bits and demonstrated a more efficient use of RS codes compared to the previous scheme and the results from [33] for face template protection.

Inner code	Outer code	FRR, %	Key, bit/frames $\times$ dimension	Efficiency
RS (63,15)	Linear (6,3,1)	1.0	$90/2 \times 63(180/4 \times 63)$	0.119
RS (63,15)	REP (3,1,1)	0.5	$90/3 \times 63(180/3 \times 63)$	0.0079
RS (31,9)	REP (3,1,1)	0.5	$90/6 \times 31$	0.0968
RS (63,21)	—	0.7	$126/1 \times 63$	0.33
RS (31,17)	—	0.3	$170/2 \times 31$	0.5
RS (31,17)	REP (3,1,1)	<0.1	$170/6 \times 31$	0.1828

**Table 2.**  
*Evaluation of FRR and key size for different ECC structures.*

For all studied schemes, privacy leakage was assessed. The calculated mutual information between the input (output) data was significantly less than the entropy of the ECC codewords, which actually confirms the impossibility of compromising the user biometric data.

5. Conclusion

Despite the fact that the development of error-correcting codes was aimed at application in communication systems, their use is also relevant in security systems, where it is required to neutralize the noise added to the data from the environment. In this chapter, the main code structures that have found application in the flow watermarking for network intrusion detection, as well as in biometric authentication systems, have been considered.

The watermarking environment model is treated as a channel with substitution, insertion, and deletion errors. Two main code constructions were considered, first: based on HMM with adding a synchronizing key sequence to sparse data and second: based on the use of the modified error-correcting VT codes with a marker attachment. Statistical and computational experiments have shown the same performance of these schemes in terms of watermark detection  $TPR \approx 1$  when  $FPR < 1\%$  with a simpler implementation of the second scheme, which is slightly inferior in coding rate to the first one. At the same time, the considered implementations of FW schemes are invisible and sufficiently resistant to network artifacts if their relative values do not exceed 20%.

In addition, two types of face biometric authentication systems based on HOG structures and latent autoencoder data were considered. The fuzziness of the HOG data was compensated by using binary BCH codes (511,58) and (511,28), which made it possible to obtain the FRR parameter value of 3.5%. The use of non-binary turbo

codes of rate  $1/3$  with octal data modulation provided the possibility to improve performance up to the value of  $FRR = 1\%$  with real helper data. And the use of concatenated RS codes together with linear binary codes showed the possibility of increasing efficiency and achieving FRR values of less than  $1\%$ . Moreover, it has been shown that a decrease in the FRR parameter is possible, firstly, by increasing the redundancy of the concatenated ECC, and secondly, by using the additional information from helper data when exploiting the EED for RS codes.

Thus, the transition to efficient non-binary code structures and real-valued ECC is a promising area of research in the field of watermarking and biometrics.

## Acknowledgements

The authors express their gratitude to the publisher for financial support.

## Conflict of interest


The authors declare no conflict of interest.

## Author details

Boris Assanovich\*, Iryna Korlyukova and Andrei Khombak  
Yanka Kupala State University of Grodno, Grodno, Belarus

\*Address all correspondence to: [bas@grsu.by](mailto:bas@grsu.by)

## IntechOpen

© 2023 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 



## References

- [1] Qin Z, Ye H, Li GY, Juang B-HF. Deep learning in physical layer communications. *IEEE Wireless Communications*. 2019;**26**(2):93-99. DOI: 10.1109/MWC.2019.180060
- [2] Nachmani E, Marciano E, Lugosch L, Gross WJ, Burshtein D, Y. Be'ery. Deep learning methods for improved decoding of linear codes. *IEEE Journal of Selected Topics in Signal Processing*. 2018;**12**(1): 119-131. DOI: 10.1109/JSTSP.2017.2788405
- [3] Huang L, Zhang H, Li R, Ge Y, Wang J. AI coding: Learning to construct error correction codes. *IEEE Transactions on Communications*. 2020;**68**(1):26-39. DOI: 10.1109/TCOMM.2019.2951403
- [4] Morelos-Zaragoza RH. *The Art of Error Correcting Coding*. 2nd ed. Chichester, West Sussex, England: Wiley; 2006. p. 278. DOI: 10.1002/0470035706
- [5] Sklar B. *Digital Communications: Fundamentals and Applications*. 2nd ed. Upper Saddle River, N.J.: Prentice-Hall PTR; 2001. p. 1079
- [6] Assanovich B, Kosarava K. Authentication System Based on Biometric Data of Smiling Face from Stacked Autoencoder and Concatenated Reed-Solomon Codes. PRIP'2021. CCIS, 1562. In: *Proceedings of the 15th International Conference*. Cham: Springer; 2021. pp. 205-219
- [7] Guruswami V, Rudra AM. *Sudan Essential Coding Theory*. University at Buffalo; 15 Mar 2019. p. 473. eBook (Creative Commons Licensed, 2022)
- [8] Carrasco RA, Johnston M. Non-Binary Error Control Coding for Wireless Communication and Data Storage. Chichester, West Sussex, United Kingdom: Wiley; 2008. p. 322
- [9] Varshamov RP, G.M. Tenengol'ts. Correction code for single asymmetric errors. *Avtomat. Telemekh*. 1965;**26**(2): 286-290
- [10] G. M. Tenengol'ts. Class of codes correcting bit loss and errors in the preceding bit. *Avtomat. Telemekh*. 1976; **37**(5):797-802
- [11] Levenshtein VI. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Doklady*. 1966; **10**(8):707-710
- [12] Abdel-Ghaffar KAS, Ferreira HC. Systematic encoding of the Varshamov-Tenengolts codes and the Constantin-Rao codes. *IEEE Transactions on Information Theory*. 1998;**44**:340-345
- [13] Abdel-Ghaffar KAS, Ferreira HC, Cheng L. Correcting deletions using linear and cyclic codes. *IEEE Transaction on Information Theory*. 2010;**56**(10): 5223-5234
- [14] Assanovich B, Puech W, Tkachenko I. Use of linear error-correcting subcodes in flow watermarking for channels with substitution and deletion errors. In: *Proceedings 14th IFIP TC 6/TC 11 Int. Conf. Commun. Multimedia Security (CMS)*. Magdeburg, Germany; 2013. pp. 105-112
- [15] Abroshan M, Venkataramanan R, Fabregas AGI. Efficient systematic encoding of non-binary VT Codes. In: *2018 IEEE International Symposium on Information Theory (ISIT)*. Vail, CO, USA: IEEE; 17-22 Jun 2018. pp. 91-95. DOI: 10.1109/ISIT.2018.8437715

- [16] Wang X, Reeves DS, Wu SF, Yuill J, Sleepy watermark tracing: An active network-based intrusion response framework. In: Proceedings Trusted Information New Decade Challenge IFIP TC11 16th Annu. Working Conference Information. Paris, France: Security (IFIP/SEC); 2001. pp. 369-384
- [17] Gong X, Rodrigues M, Kiyavash N. Invisible flow watermarks for channels with dependent substitution and deletion errors. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Proc. Kyoto, Japan; 2012. pp. 1773-1776
- [18] Gong X, Rodrigues M, Kiyavash N. Invisible flow watermarks for channels with dependent substitution, deletion, and Bursty insertion errors. IEEE Transactions on Information Forensics and Security. 2013;8(11): 1850-1859
- [19] Davey MC, Mackay DJC. Reliable communication over channels with insertions, deletions, and substitutions. IEEE Transactions on Information Theory. 2001;47(2):687-698
- [20] Yazdani R, Ardakani M. Reliable communication over non-binary insertion/deletion channels. IEEE Transactions on Communications. 2012; 60(12):3597-3608
- [21] Assanovich B, Terre VA, Penaranda-Foix FL. Watermarking pattern recognition in channels with substitution and bursty insertion and deletion errors. In: Proceedings of Pattern Recognition and Information Processing. Minsk; 2016. pp. 185-189
- [22] Cheng L, Swart TG, Ferreira HC, Abdel-Ghaffar KA, Codes for correcting three or more adjacent deletions or insertions. In: IEEE International Symposium on Information Theory (ISIT). Honolulu, USA; Jul. 2014. pp. 1246-1250
- [23] Tran QN, Turnbull BP, Hu J. Biometrics and privacy-preservation: How do they evolve? IEEE Open Journal of the Computer Society. 2021;2:179-191. DOI: 10.1109/OJCS.2021.3068385
- [24] Ratha NK, Connell JH, Bolle RM. Enhancing security and privacy in biometrics-based authentication systems. IBM Systems Journal. 2001;40: 614-634
- [25] Juels A, Wattenberg M. A fuzzy commitment scheme. In: Proceedings 6th ACM Conference Computer and Communications Security. Singapore; 2-4 Nov 1999. pp. 28-36. DOI: 10.1145/319709.319714
- [26] Kevenaar TAM, Schrijen GJ, van der Veen M, Akkermans AHM, Zuo F. Face recognition with renewable and privacy preserving binary templates. In: Fourth IEEE Workshop on Automatic Identification Advanced Technologies (AutoID'05). Buffalo, NY, USA; 2005. pp. 21-26. DOI: 10.1109/AUTOID.2005.24
- [27] Assanovich B, Veretilo Y. Biometric database based on HOG structures and BCH codes. In: Proceedings of Information Technology and Systems (ITS2017). Minsk; 2017. pp. 286-287
- [28] Baltrušaitis T, Robinson P, Morency L-P, OpenFace: An open source facial behavior analysis toolkit. In: 2016 IEEE Winter Conference on Applications of Computer Vision (WACV). Lake Placid, NY, USA; 2016. pp. 1-10. DOI: 10.1109/WACV.2016.7477553
- [29] Assanovich B, Veretilo Y. Fuzzy secure sketch biometric scheme based on non-binary turbo codes. In: Proceedings

of Information Technology and Systems.  
(ITS2018). Minsk; 2018. pp. 186-187

[30] Assanovich B. Application of Turbo codes for data transmission in UWB using PSK modulated complex wavelets, In. Signal Processing Workshop (SPW). 2020;2020:40-43. DOI: 10.23919/SPW49079.2020.9259136

[31] Maiorana E, Blasi D, Campisi P. Biometric template protection using Turbo codes and modulation constellations. IEEE WIFS. 2012;2012: 25-30

[32] Dibeklioglu H, Salah AA, Gevers T. Recognition of genuine smiles. IEEE Transactions on Multimedia. 2015;17(3): 279-294

[33] Chen L et al. Face template protection using deep LDPC codes learning. IET Biometrics. 2019;8:190-197