Article

Robust Code Constructions Based on Bent Functions and Spline Wavelet Decomposition

Alla Levina *,† and Gleb Ryaskin †

Faculty of Computer Science and Technology, Saint Petersburg Electrotechnical University "LETI", Professora Popova Str. 5, 100190 Saint-Petersburg, Russia
* Correspondence: alla_levina@mail.ru
† These authors contributed equally to this work.

Abstract: The paper investigates new robust code constructions based on bent functions and spline–wavelet transformation. Implementation of bent functions in code construction increases the probability of error detection in the data channel and cryptographic devices. Meanwhile, the use of spline wavelet theory for constructing the codes gives the possibility to increase system security from the actions of an attacker. Presented constructions combine spline-wavelets functions and bent functions. Developed robust codes, compared to existing ones, have a higher parameter of the maximum error masking probability. Illustrated codes ensure the security of transmitted information. Some of the granted constructions were implemented on FPGA.

Keywords: robust codes; bent-functions; spline-wavelet decomposition; error detection

MSC: 11T71; 41A15 94B60; 94B40; 68P30; 94A60

Citation: Levina, A.; Ryaskin, G. Robust Code Constructions Based on Bent Functions and Spline Wavelet Decomposition. *Mathematics* **2022**, *10*, 3305. https://doi.org/10.3390/math10183305

Academic Editor: Antanas Cenys

Received: 13 July 2022
Accepted: 7 September 2022
Published: 12 September 2022

1. Introduction

Nowadays, the volume of processed information is constantly growing, and it is necessary to pay due attention to the security and immutability of transmitted and stored information. The most used method of protecting information is the use of error detection and correction codes. However, hardware implementations of error correction codes, data storage systems, and cryptographic algorithms are vulnerable to malicious analyses [1,2]. The class of attacks that exploit the physical properties and peculiar properties of system architecture is called side-channel attacks [3]. Side-channel attacks are one of the most effective ways to breach the security of information, this class of attacks uses vulnerabilities in the implementation of the algorithm to obtain secrets [3–5]. Analysis of system behavior in case of its incorrect work can give to an attacker a great advantage and valuable information [3–5].

An attacker can exercise various effects on the hardware component of a cryptographic device to distort information at some stages of coding [6,7]. Additionally, an attacker has the capability of adjusting the fault injection mechanisms and techniques to inject errors of almost any multiplicity and any type. An attacker can even change the information transmitted over the channel and do it in such a way that it will be undetected by protective systems. This type of attack is called a calculation error attack [1,6]. The model of calculation error attack over an abstract storage device was first described by Cramer et al. in [1]. This type of attack poses a serious threat to the integrity of information since successful attacks can be used to outflank a protection mechanism [1]. This work also presented codes that can protect the system from such attacks—algebraic manipulation detection (AMD) codes. Accordingly, traditional error checking mechanisms are not suitable, since the scenario in which the attack develops can be atypical.

To protect systems from this type of attack, robust codes built on non-linear functions are used, because linear functions do not show all errors due to linear properties [8]. The most promising non-linear functions with higher non-linearity are bent functions [9].

Another important task for ensuring a high level of work for security code creation of a model, in which the encoding time will be minimal, without reducing the protective functions [10,11].

A design of AMD and robust codes was first proposed to protect two-level and three-level NAND flash memory for SSD drives. Due to its high data density, NAND flash memory used in SSDs is characterized by errors of various multiplicity, which directly affect security and performance. The use of hashing or digital signature algorithms is not suitable for flash memory, since the memory changes quickly, the volumes of information are very large, and the number of calculations will greatly slow down the operation of devices, unlike codes that everyone calculates on the fly. Additionally, one of the ways to use these codes is to protect cryptographic primitives from memory changes through algebraic manipulations, as well as to keep public-key encryption resilient against related-key attacks [12]. Another topical trend in AMD code theory is the protection of implantable medical devices (IMD) from algebraic manipulations, in which errors can cause direct harm to human health [13].

This article investigates the properties of robust codes constructed on bent functions and wavelet decomposition.

The paper is organized as follows. Section 2 introduces the terminology used throughout this paper. Section 3 describes the robust code, based on spline-wavelet decomposition and bent functions. Section 4 describes the implementation of constructed codes for FPGA, comparing them to each other. Section 5 describes hardware architectures. Finally, Section 6 presents the conclusions.

2. Theoretical Assumptions

The linear codes that are used in most protocols and data transfer standards are not suitable for protection against algebraic manipulation attacks [1] since one can always choose an error that will not be detected by the receiver. The general model of algebraic manipulation (including weak and strong conditions) over an abstract storage device was first presented by Cramer et al. [1] and is demonstrated in Figure 1.

Definition 1. *AMD codes. Let m and n be two positive integers. An (m,n) AMD code is a pair of a probabilistic encoding function $E : S \to G$ from a set S of size m into a finite Abelian group G of order n, and a deterministic decoding function $D : G \to S \cap \{\perp\}$ such that $D(E(s)) = s$ with probability 1 for every $s \in S$, where $\perp$ denotes combinations which are not included in the code. An AMD code is called "systematic" if set S is a group and the encoding function E has the form*

$$E : S \to S \cdot G_1 \cdot G_2$$

$$s \to (s, t, F(t,s)),$$

for a function F, with t being randomly chosen with uniform probability in G_1.

Any protected device in that model is given as an abstract storage device $\sum(G)$ which can hold an element g from a finite Abelian group G. For both strong and weak conditions, an attacker does not know the element g and can change the stored element g only by adding error $e \in G$. Successful error injection is called an algebraic manipulation. In the weak model, the attacker can choose the value of e, but cannot change the value of s and therefore cannot influence the probability of certain input combinations occurring. In the case of a strong attack, the attacker can influence the exits by choosing the inputs. In this case, the attacker knows the value of $s \in S$.

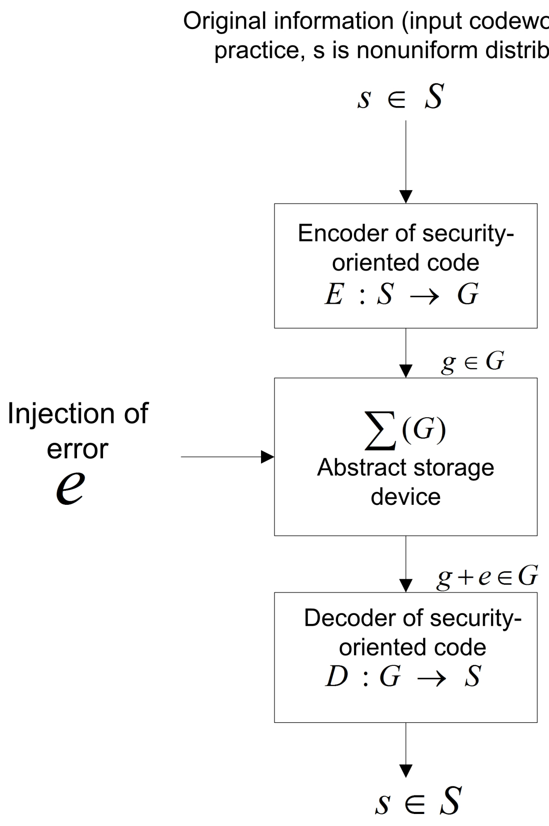
Original information (input codewords). In practice, s is nonuniform distributed
$s \in S$
Encoder of security-oriented code
$E : S \to G$
$g \in G$
Injection of error
e
$\sum(G)$
Abstract storage device
$g + e \in G$
Decoder of security-oriented code
$D : G \to S$
$s \in S$

Figure 1. Algebraic manipulation model.

To solve the problem of algebraic manipulation, in 2007 Mark Karpovsky proposed the use of robust codes, which are related to weak AMD codes [6], and this class of codes was actively used to ensure a high level of information security [13].

Definition 2. *Robust codes are non-linear systematic error-detecting codes that provide uniform protection against all errors without any (or minimize) assumptions about the error and fault distributions, capabilities, and methods of an attacker [6].*

Let $M = \|C\|$, this is the number of codewords in code C. By the definition of an $R - robust\ code$, there are no more than R code words that cannot be detected for any fixed error e.

$$R = max \parallel \{x \| x \in C, x + e \in C\} \parallel$$

The probability of masking any error can be defined as:

$$Q(e) = \frac{\{x \parallel x \in C, x + e \in C\}}{M} \tag{1}$$

In the case of linear codes, it is possible to choose an error whose the masking probability will be equal to 1, and the task of constructing a reliable code is to create the code with minimal masking probability over the entire space of errors. Therefore, the maximum error masking probability is the most important parameter for protecting information in a channel or data storage device and can be defined as

$$Q(e) = max\ \frac{\{x \parallel x \in C, x + e \in C\}}{M} = \frac{R}{M} \tag{2}$$

A graphic depiction of the definitions of the properties of a robust code is demonstrated in Figure 2.

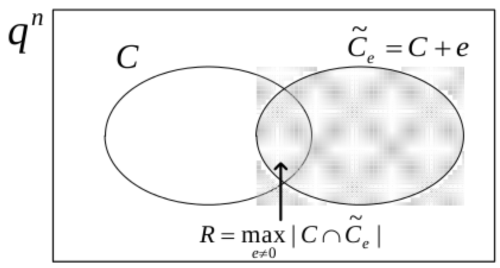

Figure 2. Definition of robustness.

In order to avoid linear properties, robust codes are constructed on the non-linear functions, therefore the parameters of a robust code are dependent on the non-linear properties of the used functions. Under such conditions, the most interesting are non-linear functions for which the property of non-linearity is maximum [14]. One such function is a bent function.

Definition 3. *A bent function is a Boolean function with an even number of variables for which the Hamming distance from the set of affine Boolean functions with the same number of variables is maximal [15].*

The bent function can be defined as a function that is extremely poorly approximated by affine functions. For the first time, bent functions were researched by O. Rothhouse in the middle of the 20th century, then they are also mentioned by J. Dillon and R.L. MacFarland [9]. Currently, the research on bent functions is widespread, however, many questions remain in this subject unexplored and require careful consideration.

Definition 4. *The non-linearity of a function f is the distance from f to a class of affine functions. Let us denote the non-linearity of the function f in terms of*

$$N_f : N_f = d(f, A(n)) = min\, d(f, g),$$

where $A(n)$ is the class of linear functions, $g \in A(n)$.

The function $f \in P_2(n)$ is called maximally non-linear if $N_f = 2^{n-1} - 2^{\frac{n}{2}-1}$ [16].

The properties of bent functions:

1. Bent functions exist only for even n;
2. Bent functions are not balanced;
3. Bent functions depend statistically on all their arguments;
4. Let f be a bent function, and h belong to the class of linear functions. Then $f + h$ belongs to the class of bent functions;
5. Let $f \in P_2(n), g \in P_2(m)$ — be functions of disjoint sets of variables. Then $f + g$ is a bent function if and only if f and g are bent functions.

For example, the Kerdock code [6] is a robust code, and at the same time is a bent function using the above properties:

$$\mathbf{x_{2s+1} = x_1 \cdot x_2 + x_3 \cdot x_4 + \cdots + x_{2s-1} \cdot x_{2s}}$$

From the cryptographic point of view, the important criteria that a Boolean function f of n variables must satisfy are the following:

1. Equilibrium—the function f takes values 0 and 1 equally often;

2. The propagation criterion $PC(k)$ of order k—for any non-zero vector $y \in Z_2^n$ weight at most k, the function $f(x+y) + f(x)$ is balanced;
3. the maximum non-linearity—the function f is such that the value of its non-linearity N_f is maximal.

The bent function matches the criteria propagation and maximum non-linearity, it allows it to detect all errors in the channel and to have a uniform probability of detecting errors. For example, the Kerdock code detects all errors in the channel due to maximum non-linearity with parameter $Q(e) = 0.5$. The propagation criterion follows from the criteria of maximum non-linearity and protects the systems against error selection, which can be made by an attacker, each error will be detected after a certain number of injections. However, bent functions are not balanced [8,9]. The equilibrium criteria are important in the case of using bent functions in cryptographic S-Boxes, but in coding theory this is not a necessary criterion, since it does not affect the robust parameters, therefore, in the framework of this study, this property was not observed.

3. Spline Wavelet Code with Different Degrees on Bent Function

One of the most well-known methods of dividing information sets into streams is wavelet transformation [17–19]. This method is used in many fields of science, including error protection codes [6,8,17,20–26]. In this article will be used wavelet transformation or rather first-degree spline-wavelet transformation for creating a robust code [25,26].

Definition 5. *Let us take function $s(t)$, where t belong to the Hilbert space $L^2(R)$ with the scalar product $<f(t), g(t)> = \int_{-\infty}^{+\infty} f(t)g(t)dt$ and the norm $\int_{-\infty}^{+\infty} |s(t)|^2 dt < \infty$.*

The idea of the wavelet transform is based on the partition of the signal $s(t)$ into two components, approximating $A_m(t)$ and detailing $D_m(t)$, where m denotes the decomposition (reconstruction) level.

$$s(t) = A_m(t) + \sum_{i=1}^{m} D_i(t)$$

In this article, will be used spline wavelet transformation for the creation of an error detection code.

Let X be a non-uniform grid of elements, $X = \{x_j\}$, where $j \in Z$, Z is the set of integers. First-degree splines on the grid X are defined as follows:

$$\omega_j(t) = \begin{cases} (t - x_j)(x_{j+1} - x_j)^{-1} & t \in [x_j, x_{j+1}) \\ (x_{j+2} - t)(x_{j+2} - x_{j+1})^{-1} & t \in [x_{j+1}, x_{j+2}) \\ 0 & t \notin [x_j, x_{j+2}) \end{cases}$$

As a part of this research, we considered spline wavelets of a higher degree, but this leads to an increase in mathematical operations without any benefit. Other types of wavelets did not suit us due to the impossibility of converting them to binary form.

In the process of wavelet decomposition, an element x_k is taken out from the grid X, after this transformation will be received the new grid $\tilde{X}$.

$$\tilde{x}_j = x_j, j \leq k-1,$$
$$\tilde{x}_j = x_{j+1}, j \geq k,$$
$$\epsilon = x_k$$

where $j \in Z$, Z is the set of integers.

Based on this new grid new splines $\widetilde{\omega_j}(t)$ are constructed. New and old splines are interconnected. The relationship between the elements $\omega_j(t)$ and $\widetilde{\omega_j}(t)$) can be shown using the formulas. The new splines $\widetilde{\omega}_j(t)$ depend on the old $\omega_j(t)$ as follows:

$$\widetilde{\omega}_j(t) = \omega_j(t), j \leq k-3$$
$$\widetilde{\omega}_j(t) = \omega_{j+1}(t), j \geq k,$$
$$\widetilde{\omega}_{k-2}(t) = \omega_{k-2}(t) + \widetilde{\omega}_{k-2}(x_k)\omega_{k-1}(t),$$
$$\widetilde{\omega}_{k-1}(t) = \omega_k(t) + \widetilde{\omega}_{k-1}(x_k)\omega_{k-1}(t)$$

In this research, the algorithm of spline wavelet decomposition will be used for building robust codes. Let us rewrite the spline-wavelet decomposition formula in terms of code creation:

$$\mathbf{Wave_k} = \mathbf{c_k} - \mathbf{c_{k+1}} - (\mathbf{x_{k+2}} - \mathbf{x_{k-1}})(\mathbf{x_{k+2}} - \mathbf{x_k})^{-1}(\mathbf{c_{k-1}} - \mathbf{c_{k+1}}) \tag{3}$$

where $1 \leq j \leq (n-k)/2$, k—the number of characters in code, c_k—informational element, $z_i \in Z$.

Spline-wavelet decomposition elements will be used for the construction of redundant symbols of robust codes. This will allow the creation of a large class of robust codes depending on the spline-wavelet grid values. In the case of changing the grid, without changing the construction, it is possible completely to change the properties of a robust code, for example, increasing the parameter R and decreasing the probability of masking any error. Without this method, in case of using only AMD code, the attacker can adapt to the system and find weaknesses.

For example, when the system is implementing a combination of LDPC code [27] and AMD code for protection against algebraic-manipulation attacks (the structure is shown in Figure 3), knowledge about the sparseness of the check matrix in LDPC gives an attacker significant advantages in both increasing the error masking probability and simplifying the process of finding undetectable errors [27]. For a successful algebraic manipulation, an attacker needs to make an error simultaneously in the LDPC codeword, bypassing the check-in part of the AMD code [27]. Therefore, if there is an algorithm that will change the values of the grid, the security of the code device from a "smart" attacker will increase.

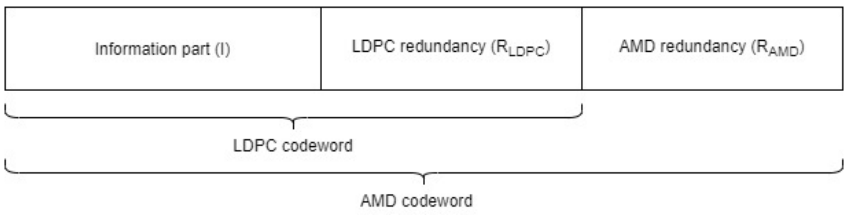

Figure 3. Structure of the combined LDPC and AMD codeword.

Constructions presented in this work will be compared by the parameter R. In this research, the value $n = 8$ is taken since most protocols work with a given number of information symbols. For each number of variables, bent functions were constructed on variables and spline-wavelet decomposition elements. In this construction, for all code, selected grid is $x = \{x_1, x_2, \ldots, x_{n-1}, x_k\}$. It is based on static values, or on the information part of the codeword.

The grid is very important factor for robust codes, because if the grid is static, the wavelet element will became a simple affine function: $Wave_k = c_k - c_{k+1} - (x_{k+2} - x_{k-1})(x_{k+2} - x_k)^{-1}(c_{k-1} - c_{k+1})$, where $(x_{k+2} - x_{k-1})(x_{k+2} - x_k)^{-1}$ is 0 or 1, but if the grid is based on information part, wavelet element become a non-linear function of second or third degree.

Let $c = \{c_1, c_2, \ldots, c_{n-1}, c_n\}$ denotes the codeword of some shared (n, k) code. Then $\{c_1, c_2, \ldots, c_{k-1}, c_k\}$ is the information part, and $c_{k+1}, \ldots, c_n$—additional part, $n = k + 2$. For presented construction grid is selected depending on the spline wavelet function, $f_i(c_1, c_2, \ldots, c_{k-1}, c_m)$ is a function from Table 1, $m = 8$, m is the number of parameters of the function f, it can correspond to n or can be less than n if we use the function from Table 1 with the number of variables $n > 8$. The vector c belongs to the code if

$$\mathbf{c_{k+1} = f_0(c_1, \ldots, c_m) + c_{m+1} \cdot c_{m+2} + \ldots + c_{k-1} \cdot c_k;}$$
$$\mathbf{c_{k+2} = f_1(c_1, \ldots, c_m) + c_{m+1} \cdot c_{m+2} + \ldots + c_{k-1} \cdot c_k;}$$

The results of the analysis designed code show that the parameter R and, accordingly, the maximum probability of hiding the error $Q(e)$ is lower than that of the existing solutions with the same number of additional bits. However, the designed solution also has disadvantages—the time for encoding information can be quite high [25].

Functions for n = 8 are presented in Table 1, which also gives the conditions of the grid and the degree of function.

Table 1. Spline wavelet bent functions for n = 8.

Number of Function	Grid	Function	Deg(f)
1	$x_i = c_i$	$f_i = c_{i+1} \cdot c_{i+3} \cdot c_{i+4} + c_{i+2} \cdot c_{i+3} \cdot c_{i+5} + Wave_{i+2} \cdot c_{i+6} + c_i \cdot c_{i+3} + c_i \cdot c_{i+5} + c_{i+2} \cdot c_{i+3} + c_{i+2} \cdot c_{i+4} + c_{i+2} \cdot c_{i+5} + c_{i+3} \cdot c_{i+4} + c_{i+3} \cdot c_{i+5} + c_{i+6} \cdot c_{i+7}$	4
2	Static	$f_i = c_i \cdot c_{i+1} \cdot Wave_{i+2} + c_{i+1} \cdot c_{i+3} \cdot Wave_{i+4} + c_i \cdot c_{i+1} + c_i \cdot c_{i+3} + c_{i+1} \cdot c_{i+5} + c_{i+2} \cdot c_{i+4} + c_{i+3} \cdot c_{i+4} + c_{i+6} \cdot c_{i+7}$	3
3	$x_i = c_{7-i}$	$f_i = c_i \cdot c_{i+1} + Wave_{i+2} \cdot c_{i+3} + c_{i+4} \cdot c_{i+5} + c_{i+6} \cdot c_{i+7}$	4
4	Static	$f_i = c_i \cdot Wave_i + c_{i+1} \cdot Wave_{i+2} + c_{i+2} \cdot Wave_{i+4} + c_i \cdot c_{i+3} + c_{i+1} \cdot c_{i+5} + c_{i+2} \cdot c_{i+3} + c_{i+2} \cdot c_{i+4} + c_{i+2} \cdot c_{i+5} + c_{i+3} \cdot c_{i+4} + c_{i+3} \cdot c_{i+5} + c_{i+6} \cdot c_{i+7}$	2

Let us compile the code constructions for all the above functions with two redundant symbols $r_0 = f_0$, $r_1 = f_1$ for $n = 8$. Calculated parameter R and the maximum probability of error concealment, the results are listed in Table 2. A comparison was also made with the Kerdock code, which is the most well-known and used robust code. Additionally, Table 2 shows the time taken to encode 5000 bytes of information calculated by the software. The value of 5000 bytes is taken to represent the encoding time and to understand the difference between the compared codes.

Table 2. Code constructions comparison for developed functions.

Function	The Degree of Bent Function	R	Maximum Probability of Error Concealment, Q(e)	The Average Value, s
Function 1	4	96	0.375	0.067
Function 2	3	120	0.46875	0.059
Function 3	4	96	0.375	0.066
Function 4	2	128	0.5	0.054
Kerdock code	2	128	0.5	0.049

Lemma 1. *With an increase in the number of information symbols, the function stays a bent function.*

Proof. Since when adding multiplicative elements $c_{n+1} * c_{n+2}$, by property 5 of the bent function, the final function will also be a bent function. □

One of the main parameters of code constructions is coding time, but at the same time, code constructions need to be created with a high degree of bent functions, because, for security reasons, we need to lower the maximum probability of error concealment. The solution for it is based on matrices and optimization of hardware implementation.

4. FPGA Implementation of Presented Constructions

In this section, we will show the FPGA implementation of the presented constructions, and demonstrate its matrices representation. Let us try to roughly represent the created constructions in the form of matrix multiplication, to identify new constructions, and further optimization of the algorithms for the (10,8) code. Selected designs are built based on a more convenient implementation for FPGA—the minimum number of connections and calculations FPGA implementation of presented constructions.

For all constructions, the spline-wavelet function is the same and can be described as:

$$Wave_k = c_k - c_{k+1} - (x_{k+2} - x_{k-1})(x_{k+2} - x_k)^{-1}(c_{k-1} - c_{k+1})$$

It is possible to use linear components for additional methods of construction change.

Lemma 2. *The use of linear components for an additional method of structural change does not reduce the non-linear properties of the functions.*

Proof. According to property 4 of the bent functions, the linear component does not affect the non-linear properties. □

4.1. Construction 1

The grid x is static and can be selected as desired. The matrix function of the code is presented below:

$$\begin{pmatrix} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{Wave_0} & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{Wave_1} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \mathbf{Wave_2} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \mathbf{Wave_3} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \mathbf{Wave_4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \mathbf{Wave_5} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \mathbf{Wave_6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \mathbf{Wave_7} \end{pmatrix} =$$
$$= \begin{pmatrix} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & r_0 & r_1 \end{pmatrix}$$

The following is the classic form:

$$\mathbf{r_0} = \mathbf{c_0} \cdot \mathbf{Wave_0} + \mathbf{c_2} \cdot \mathbf{Wave_2} + \mathbf{c_4} \cdot \mathbf{Wave_4} + \mathbf{c_6} \cdot \mathbf{Wave_6}$$
$$\mathbf{r_1} = \mathbf{c_1} \cdot \mathbf{Wave_1} + \mathbf{c_3} \cdot \mathbf{Wave_3} + \mathbf{c_5} \cdot \mathbf{Wave_5} + \mathbf{c_7} \cdot \mathbf{Wave_7}$$

All functions are bent functions of degree 2 constructed from spline wavelets and Kerdock code. Its FPGA implementation is shown in Figure 4.

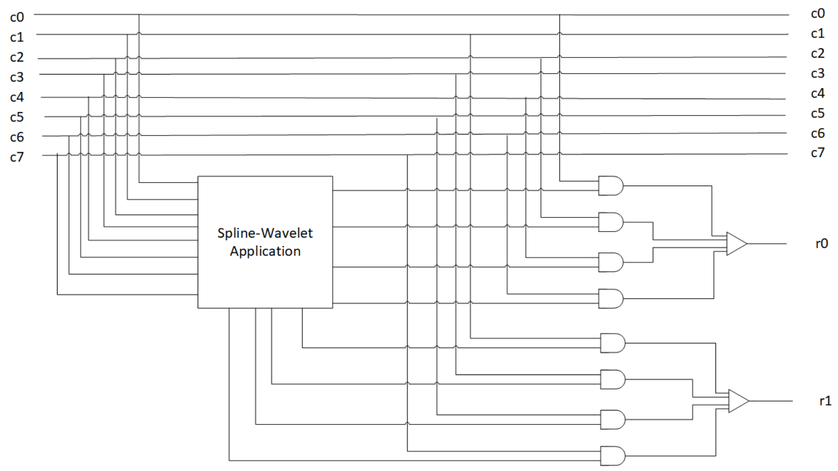

Figure 4. Hardware architecture for construction.

4.2. Construction 2

The grid $x : x_i = c_{i-1}$. The matrix function of the code is presented below:

$$\begin{pmatrix} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{Wave_1} & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{Wave_2} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \mathbf{c_3} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \mathbf{c_4} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \mathbf{c_5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \mathbf{c_6} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \mathbf{Wave_7} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \mathbf{Wave_0} \end{pmatrix} =$$

$$= \begin{pmatrix} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & r_0 & r_1 \end{pmatrix}$$

The following is the classic form:

$$\mathbf{r_0} = \mathbf{c_0} \cdot \mathbf{Wave_1} + \mathbf{c_2} \cdot \mathbf{c_3} + \mathbf{c_4} \cdot \mathbf{c_5} + \mathbf{c_6} \cdot \mathbf{Wave_7}$$
$$\mathbf{r_1} = \mathbf{c_1} \cdot \mathbf{Wave_2} + \mathbf{c_3} \cdot \mathbf{c_4} + \mathbf{c_5} \cdot \mathbf{c_6} + \mathbf{c_7} \cdot \mathbf{Wave_0}$$

All functions are bent functions of degree 3 constructed from spline wavelets and Kerdock code. Its FPGA implementation is shown in Figure 5.

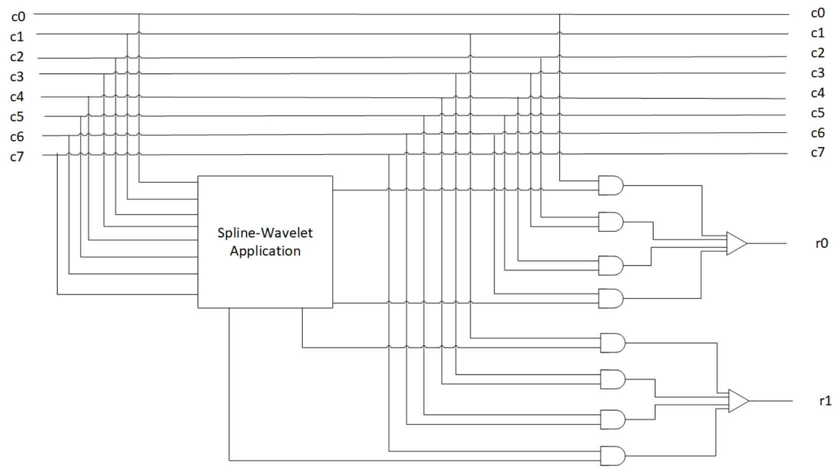

Figure 5. Hardware architecture for construction 2.

4.3. Construction 3

The grid x: $x_i = c_{i-1}$. The matrix function of the code is presented below:

$$\begin{pmatrix} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c_1} & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c_2} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \mathbf{Wave_2} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \mathbf{c_5} & \mathbf{Wave_3} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \mathbf{c_6} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \mathbf{Wave_7} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \mathbf{Wave_0} \end{pmatrix} =$$
$$= \begin{pmatrix} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & r_0 & r_1 \end{pmatrix}$$

The following is the classic form:

$$\mathbf{r_0} = \mathbf{c_0} \cdot \mathbf{c_1} + \mathbf{Wave_2} \cdot \mathbf{c_3} + \mathbf{c_4} \cdot \mathbf{c_5} + \mathbf{c_6} \cdot \mathbf{Wave_7}$$
$$\mathbf{r_1} = \mathbf{c_1} \cdot \mathbf{c_2} + \mathbf{Wave_3} \cdot \mathbf{c_4} + \mathbf{c_5} \cdot \mathbf{c_6} + \mathbf{c_7} \cdot \mathbf{Wave_0}$$

All functions are bent functions of degree 4 constructed from spline wavelets and Kerdock code. Its FPGA implementation is shown in Figure 6.

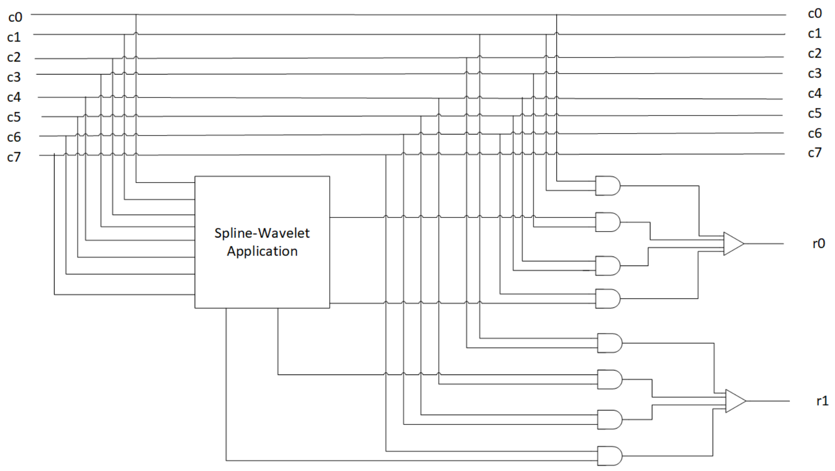

Figure 6. Hardware architecture for construction 3.

4.4. Code Construction Comparison

When comparing different codes, bit overhead (BO) is also an important parameter. Bit overhead is defined as the ratio of parity bits to data bits present in a codeword. A better error detection and correction method should have less bit overhead [28].

$$BitOverhead(BO) = Paritybits(n)/Databits(m)$$

In this research, bit overhead is the same for all compared codes. A large number of additional bits does not make much sense in a code structure, since its main task is not to correct errors, but to detect all of them, so it may be a good idea to combine robust and linear codes that will solve different problems—detecting all errors and correcting errors of certain type.

The bit overhead, parameter R, and maximum probability of error concealment for presented constructions are listed in Table 3. Additionally, Table 3 shows the time taken to encode 5000 bytes of information calculated by software.

Table 3. Code constructions comparison.

Function	The Degree of Bent Function	R	Maximum Probability of Error Concealment, Q(e)	The Average Value, s	Bit Overhead
Construction 1	2	128	0.5	0.048	0.25
Construction 2	3	120	0.46875	0.054	0.25
Construction 3	4	96	0.375	0.056	0.25
Kerdock code	2	128	0.5	0.049	0.25
Robust Duplication Code ($f(x) = x^2$)	-	2	0.0078125	0.24	1

A degree of the bent function different from 2 gives a better result for the parameter R. The number of calculations and the time spent on coding information is more compared to the codes built on bent functions with a power of 2. When these codes are used in the case of protection against attack, then the parameter R is more important. The best value of the R parameter has the robust duplication code, but the information encoding time is very long and the number of bits is equal to the number of information symbols, which can disrupt the operation of the device where this code can be applied. Using spline wavelets gives the possibility to build a large number of robust codes, and bent functions, and increase their degree, thereby improving the quality of robust codes. Additionally, if the number of information symbols increases for a codeword, the functions will stay bent functions.

5. General Robust Hardware Architectures

The developed method can be applied both to protect information transmitted over communication channels or to protect information on hardware devices. The general architecture used for protecting hardware devices is shown in Figure 7. The architecture is based on adding redundancy around an original device to create data redundancy which can be used to verify data integrity and the correct operation of the device. The architecture is composed of three major hardware components: original hardware, redundant hardware for predicting the *r*-bits signature of the original device, and an error-detecting network (EDN) which verifies the relationship of the output of the original device and the signature of the predictor.

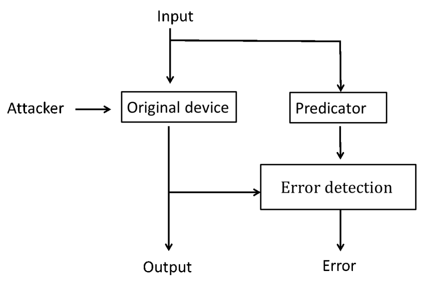

Figure 7. General architecture for the protection of hardware with error-detecting codes.

The general architecture used for protecting communication channels is shown in Figure 8. The architecture is based on adding redundancy which can be used to verify data integrity after passing through the communication channel on the receiver side. Coding information with a robust code is better to combine with a linear code in the communication lines in order to be able to correct the minimum noise.

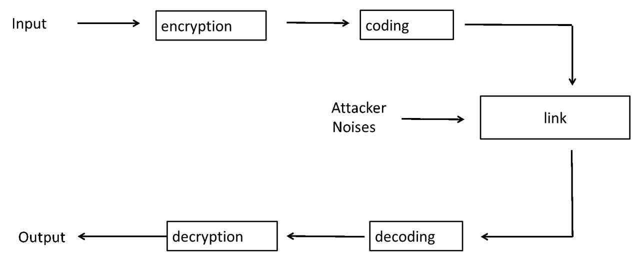

Figure 8. General architecture for the protection of communication channel with error-detecting codes.

6. Conclusions

The protection methods which have been adapted from traditional fault-tolerant architectures are not optimal for the protection of cryptographic hardware or communication channels susceptible to side-channel attacks. The robust constructions can be applied to existing architectures based on linear error-detecting codes in communication channels to increase their error-detecting power and reduce the number of undetectable errors.

In this paper, we described the error-correcting coding scheme based on wavelet transformation and bent functions. For the proposed scheme, we created spline-wavelet robust codes on bent functions. The robust wavelet code has no undetectable errors, so it ensures reliable protection against the error injection and has a lower maximum error masking probability. Designs have been developed with the least information encoding time among this class of codes, which are best for implementation in FPGA.

Author Contributions: Conceptualization, A.L. and G.R.; methodology, A.L.; software, G.R.; validation, G.R.; formal analysis, G.R.; investigation, G.R.; resources, A.L.; data curation, G.R.; writing—original draft preparation, G.R.; writing—review and editing, A.L.; visualization, A.L.; supervision, A.L.; project administration, A.L.; funding acquisition, A.L. All authors have read and agreed to the published version of the manuscript.

Funding: Ministry of Science and Higher Education of the Russian Science Foundation (Project "Goszaadanie" No. 075-01024-21-02 from 29 September 2021, FSEE-2021-0015): FSEE-2021-0015.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cramer, R.; Fehr, S.; Padro, C. Algebraic manipulation detection codes. *Sci. China Math.* **2013**, *56*, 1349–1358. [CrossRef]
2. Osnat, K.; Ilia, P. On resilience of security-oriented error detecting architectures against power attacks: A theoretical analysis. In Proceedings of the 18th ACM International Conference on Computing Frontiers, Virtual, 11–13 May 2021; pp. 229–237. [CrossRef]
3. Genkin, D.; Pachmanov, L.; Pipman, I.; Tromer, E.; Yarom, Y. ECDSA key extraction from mobile devices via nonintrusive physical side channels. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 1626–1638.
4. Tromer, E.; Osvik, D.A.; Shamir, A. Efficient cache attacks on AES, and countermeasures. *J. Cryptol.* **2010**, *23*, 37–71. [CrossRef]
5. Mael, G.; Batya, K.; Osnat, K.; Ilia, P. Error control scheme for malicious and natural faults in cryptographic modules. *J. Cryptogr. Eng.* **2020**, *10*, 321–336. [CrossRef]
6. Karpovsky, M.G.; Kulikowski, K.; Wang, Z. Robust Error Detection in Communication and Computation Channels. In *Spectral Methods and Multirate Signal Processing*; Keynote Paper; SMMSP: Moscow, Russia, 2007.
7. Genkin, D.; Pachmanov, L.; Pipman, I.; Shamir, A.; Tromer, E. Physical key extraction attacks on PCs. *ACM Commun.* **2016**, *59*, 70–79. [CrossRef]
8. Carlet, C. Boolean functions for cryptography and error correcting codes. In *Boolean Methods and Models*; Hammer, P., Crama, Y., Eds.; Cambridge University Press: Cambridge, UK, 2007.
9. Tokareva, N. *Bent Functions: Results and Applications to Cryptography*; Academic Press: Hoboken, NJ, USA, 2015.
10. Hila, R.; Osnat, K. A new class of security oriented error correcting robust codes. *Cryptogr. Commun.* **2019**, *11*, 965–978. [CrossRef]
11. Osnat, K. Security oriented codes. In Proceedings of the 2014 IEEE 28th Convention of Electrical and Electronics Engineers, Eilat, Israel, 3–5 December 2014; pp. 1–5. [CrossRef]
12. Wee, H. Public Key Encryption against Related Key Attacks. In *Public Key Cryptography—PKC 2012*; PKC 2012. Lecture Notes in Computer Science; Fischlin, M., Buchmann, J., Manulis, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7293.

13. Bu, L.; Karpovsky, M.G.; Kinsy, M.A. Bulwark: Securing implantable medical devices communication channels. *Comput. Secur.* **2019**, *86*, 498–511. [CrossRef]
14. Shao, M.; Miao, Y. Algebraic manipulation detection codes via highly nonlinear functions. *Cryptogr. Commun.* **2021**, *13*, 53–69. [CrossRef]
15. Rothaus, O.S. On "bent" functions. *J. Comb. Theory Ser.* **1976**, *20*, 300–305. [CrossRef]
16. Claude, C.; Sihem, M. Four decades of research on bent functions. *Des. Codes Cryptogr.* **2015**, *78*, 5–50. [CrossRef]
17. Fekri, F.; Mersereau, R.M.; Schafer, R.W. Theory of wavelet transform over finite fields. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing 3, Phoenix, AZ, USA, 15–19 March 1999; pp. 1213–1216.
18. Demyanovich, U.K. Minimal Splines and Wavelets. *Vestn. St. Petersburg Univ. Math.* **2008**, *41*, 88–101. [CrossRef]
19. Fekri, F.; McLaughlin, S.W.; Mersereau, R.M.; Schafer, R.W. Double circulant selfdual codes using finite-field wavelet transforms. In Proceedings of the Applied Algebra, Algebraic Algorithms and Error Correcting Codes Conference, Honolulu, HI, USA, 15–19 November 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 355–364.
20. Levina, A.; Taranov, S. Spline-wavelet robust code under nonuniform codeword distribution. In Proceedings of the Third International IEEE Computer, Communication, Control and Information Technology, Hooghly, India, 7–8 February 2015.
21. Levina, A.B.; Taranov, S.V. AMD codes based on wavelet transform. In Proceedings of the Progress in Electromagnetics Research Symposium, Singapore, 19–22 November 2017; pp. 2534–2539.
22. Levina, A.B.; Taranov, S.V. Second-order spline-wavelet robust code under nonuniform codeword distribution. *Procedia Comput. Sci.* **2015**, *62*, 297–302. [CrossRef]
23. Levina, A.B.; Taranov, S.V. Construction of linear and robust codes that is based on the scaling function coefficients of wavelet transforms. *J. Appl. Ind. Math.* **2015**, *9*, 540–546. [CrossRef]
24. Levina, A.; Taranov, S. New construction of algebraic manipulation detection codes based on wavelet transform. In Proceedings of the Conference of Open Innovation Association, FRUCT, St. Petersburg, Russia, 26–28 September 2016; art. no. 7561526, pp. 187–192.
25. Alla, L.; Gleb, R.; Igor, Z. Spline-Wavelet Bent Robust Codes. In Proceedings of the Federated Conference on Computer Science and Information Systems, Leipzig, Germany, 1–4 September 2019; pp. 227–230.
26. Levina, A.; Ryaskin, G. Implementation of Spline-Wavelet Robust Bent Code in Code-Division Multiple Access. In *Mathematics and its Applications in New Computer Systems. MANCS 2021*; Lecture Notes in Networks and Systems; Tchernykh, A., Alikhanov, A., Babenko, M., Samoylenko, I., Eds.; Springer: Cham, Switzerland, 2021; Volume 424.
27. Levina, A.; Ryaskin, G.; Taranov, S.; Polubaryeva, A. Effectiveness of Using Codes With a Sparse Check Matrix for Protection against Algebraic Manipulations. In Proceedings of the 2021 International Conference Automatics and Informatics (ICAI), Varna, Bulgaria, 30 September–2 October 2021; pp. 292–295. [CrossRef]
28. Varinder, S.; Narinder, S. Improving Performance Parameters of Error Detection and Correction in HDLC Protocol by using Hamming Method. *Int. J. Comput. Appl.* **2015**, *126*, 1–7. [CrossRef]