Reliable and Secure Memories Based on Algebraic Manipulation Correction Codes

Zhen Wang and Mark Karpovsky
Reliable Computing Laboratory, Boston University, Boston , USA
wang.zhen.mtk@gmail.com, markkar@bu.edu

***Abstract*—The reliability and security of memories are crucial considerations in the modern digital system design. Traditional codes usually concentrate on detecting and correcting errors of certain types, e.g. errors with small multiplicities or byte errors, and cannot detect or correct unanticipated errors. In this paper we present a reliable and secure memory architecture based on robust Algebraic Manipulation Correction codes. These codes can provide a guaranteed error detection probability and can correct any error regardless of its multiplicity as long as the error stays for several consecutive clock cycles. The construction and the error correction procedure for the code will be described. The probability that an error can be successfully detected and/or corrected and the hardware overhead of the memory architecture based on these codes will be estimated.**

I. INTRODUCTION

Memories are critical elements in today's digital systems. Various types of memories are widely used in many different reliable and secure applications and appear in nearly all digital devices. As the technologies move into nano realm, the reliability of memories is largely compromised by the increased rate of multi-bit errors [1]. In secure applications, the security of memories and the whole system is threatened by side-channel attacks such as fault injection attacks [2], [3].

Error control codes are often used to improve the reliability and security of memory systems. Most of the existing reliable and secure memory architectures are based on linear codes, which concentrate their error detection and correction capabilities against certain types of errors, e.g. errors with small multiplicities. The reliability and security of systems protected by linear codes largely depend on the accuracy of the expected error model. For applications where the error model is hard to predict or the multi-bit error rate is high, traditional protection methods based on linear codes are not sufficient.

As an alternative to linear codes, **robust codes** and their variants based on **nonlinear encoding functions** were proposed in [4], [5], [6]. Robust codes, however, can only provide a guaranteed level of security under the assumption that the attacker cannot control both the fault-free output of the cryptographic device and the injected nonzero errors.

To overcome the limitation of robust codes, constructions of **Algebraic Manipulation Detection (AMD)** codes based on nonlinear functions and Generalized Reed-Muller codes were proposed in [7], [8] to protect cryptographic hardware and detect faults injected by the strongest attackers. In this paper, we present Algebraic Manipulation Correction codes based on AMD codes for correcting repeating errors. The proposed codes can correct any error regardless of its multiplicity assuming it stays for several consecutive clock cycles. The error correction procedure of the proposed codes can be simplified to the problem of solving a system of linear equations combined with robust error correction described in [7].

The rest part of the paper is organized as follows. In Section II, the definitions and constructions of robust error correction codes and AMD codes are briefly introduced. We prove that AMD codes are also robust error correction codes. We describe the modification of AMD codes and present the corresponding error correction procedure for the code. The error correction capabilities of the proposed codes are simulated and analyzed. In Section III, we describe the hardware architecture of the reliable and secure memories protected by the proposed codes. Different designs for the encoder and the syndrome computation block for the proposed codes are studied and compared.

II. AMD CODES AND ROBUST ERROR CORRECTION

Throughout the paper we denote by $\oplus$ the addition in $GF(q), q = 2^r$. An AMD code V with codewords $(y, x, f(y, x))$ will be referred to as a (k, m, r) code, where $y \in GF(2^k)$ is the information part, $x \in GF(2^m)$ is the random data and $f(y, x) \in GF(2^r)$ is the redundant part.

A. Definitions, Constructions and Error Correction Properties

Definition 2.1: [9] A (k, m, r) error detecting code is called Algebraic Manipulation Detection (AMD) code iff $K_S = \{\mathbf{0}\}$, where $\mathbf{0}$ is the all zero vector in $GF(2^n)$, $n = k + m + r$ and

$$K_S = \{e | \exists y, f(y \oplus e_y, x \oplus e_x) \oplus f(y, x) \oplus e_f = \mathbf{0}, \forall x\}. \quad (1)$$

There are no undetectable errors (errors that are undetected with a probability of 1) for AMD codes. For any y and any e, the error masking probability for an AMD code V can be computed as

$$\begin{aligned} Q_V(y, e) &= 2^{-m} | \{x \,|\, (y, x, f(y, x)) \in V, \\ &\qquad (y \oplus e_y, x \oplus e_x, f(y, x) \oplus e_f) \in V\} |, \end{aligned} \quad (2)$$

which is the fraction of random m-bit vectors x that will mask a fixed error e for a given y. The security level of the system protected by AMD code can be characterized by the worst case error masking probability $Q_V = \max_y \max_{e \neq 0} Q_V(y, e)$.

Let $x = (x_1, x_2, \cdots, x_t), x_i \in GF(2^r)$. Let

$$A(x) = \begin{cases} \bigoplus_{i=1}^{t} x_i^{b+2} & \text{if } b \text{ is odd;} \\ \bigoplus_{i=2}^{t-1} x_1 x_i^{b+1} & \text{if } b \text{ is even and } t > 1; \end{cases} \quad (3)$$

The first author now works for Mediatek Wireless, Inc. The work of the second author is sponsored by the NSF grant CNR 1012910.

978-1-4673-2085-6/12/$31.00 ©2012 IEEE

where $1 \leq b \leq 2^r - 3$. Let

$$B(x,y) = \bigoplus_{1 \leq j_1+j_2+\cdots+j_t \leq b+1} y_{j_1,j_2,\cdots,j_t} \prod_{i=1}^{t} x_i^{j_i}, \quad (4)$$

where $\prod_{i=1}^{t} x_i^{j_i}$ is a monomial of x of a degree between 1 and $b+1$ and $\prod_{i=1}^{t} x_i^{j_i} \notin \Delta B(x,y)$ defined by

$$\Delta B(x,y) = \begin{cases} \{x_1^{b+1}, x_2^{b+1}, \cdots, x_t^{b+1}\} & \text{if } b \text{ is odd;} \\ \{x_2^{b+1}, x_1 x_2^b, \cdots, x_1 x_t^b\} & \text{if } b \text{ is even, } t > 1; \end{cases} \quad (5)$$

Theorem 2.1: [7] Let $f(x,y) = A(x) \oplus B(x,y)$ be a q-ary polynomial with $y \in GF(q^s)$ as coefficients and $x \in GF(q^t)$ as variables, where $1 \leq b \leq q-3$ and $q = 2^r$. Then the code V composed of all vectors $(y, x, f(x,y))$ is an (k, m, r) AMD code with $m = tr$, $k = (\binom{t+b+1}{t} - 1 - t)r$. The worst case error masking probability over all y (user defined data in the memory) and all nonzero errors e for this code is $Q_V = \max_y \max_{e \neq 0} Q_V(y,e) = (b+1)2^{-r}$.

Generally speaking, AMD codes constructed in Theorem 2.1 have a small Hamming distance and cannot be directly used for error correction. However, since the encoding operations of AMD codes are nonlinear, it is possible to conduct **robust error correction** if the error repeats for consecutive clock cycles.

Definition 2.2: [10] For any code $C \in GF(2^n)$, let $\{c(1), c(2), \cdots, c(R)\}$ be a set of R different codewords. If $\{c(1), c(2), \cdots, c(R)\}$ can be uniquely determined from

$$D_e = \{\tilde{c}(1), \tilde{c}(2), \cdots, \tilde{c}(R)\},$$

where $\tilde{c}(i) = c(i) \oplus e$, the code C is a **R-robust error correction** code. A R-robust error correction code is called **weak** if only the codewords but not their order can be derived. R-robust error correction code is called **strong** if both the codewords and their order can be derived.

AMD codes constructed in Theorem 2.1 are inherently robust error correction codes as shown in the following example.

Example 2.1: Let $k = m = r$ $(t = b = 1)$ and $f(y,x) = y \cdot x \oplus x^3, y, x, f(y,x) \in GF(2^r)$. The code V composed of all vectors $(y, x, f(y,x))$ is a AMD code with $Q_V = 2^{-r+1}$.

Let $e = (e_y, e_x, e_f), e_y \in GF(2^k), e_x \in GF(2^m), e_f \in GF(2^r)$ be the error vector. Compute the **syndrome** of the code as $S = f(\tilde{y}, \tilde{x}) \oplus f(y,x) \oplus e_f$, where $\tilde{y} = y \oplus e_y$ and $\tilde{x} = x \oplus e_x$. Suppose there are three different codewords $(y(i), x(i), f(y(i), x(i))), 1 \leq i \leq 3$ distorted by the same error $e = (e_y, e_x, e_f)$. then we have

$$\begin{aligned} S(1) &= e_x^3 \oplus e_x^2\tilde{x}(1) \oplus e_x\tilde{x}(1)^2 \oplus e_x\tilde{y}(1) \oplus e_y\tilde{x}(1) \oplus e_xe_y \oplus e_f, \\ S(2) &= e_x^3 \oplus e_x^2\tilde{x}(2) \oplus e_x\tilde{x}(2)^2 \oplus e_x\tilde{y}(2) \oplus e_y\tilde{x}(2) \oplus e_xe_y \oplus e_f, \\ S(3) &= e_x^3 \oplus e_x^2\tilde{x}(3) \oplus e_x\tilde{x}(3)^2 \oplus e_x\tilde{y}(3) \oplus e_y\tilde{x}(3) \oplus e_xe_y \oplus e_f. \end{aligned}$$

From here we have

$$e_x(\tilde{x}_{23}(\tilde{x}_{12}^2 \oplus \tilde{y}_{12}) \oplus \tilde{x}_{12}(\tilde{x}_{23}^2 \oplus \tilde{y}_{23})) = S_{12}\tilde{x}_{23} \oplus S_{23}\tilde{x}_{12}, \quad (6)$$

$$e_y\tilde{x}_{12} = e_x^2\tilde{x}_{12} \oplus e_x(\tilde{x}_{12}^2 \oplus \tilde{y}_{12}) \oplus S_{12}, \quad (7)$$

where $S_{ij} = S(i) \oplus S(j)$, $\tilde{x}_{ij} = \tilde{x}(i) \oplus \tilde{x}(j), \tilde{x}_{ij} \neq 0$ and $\tilde{y}_{ij} = \tilde{y}(i) \oplus \tilde{y}(j)$.

The correction of e_f is not necessary. If needed, e_f can be computed by substituting e_x and e_y into any of the three syndrome equations. Thereby, $e = (e_y, e_x, e_f)$ can be corrected by solving (6), (7) as long as it distorts three different codewords no matter how large its Hamming weight is.

The straightforward correction of repeating errors occurring to AMD codes can be computationally prohibitive as t or b increases. In the next Theorem, we show that by adding extra redundant bits $g(x)$ to make $\{(x, g(x))\}$ a robust error correction code, the correction of repeating errors for AMD codes can be split into two steps. In the first step, errors in x (and $g(x)$) are corrected using robust error correction algorithm shown in [10]. In the second step, errors in y (and $f(y,x)$) are corrected by solving a system of linear equations. When $\{(x, g(x))\}$ is a robust code in a special form, e.g. $g(x) = x^3$ or $g(x) = x^{-1}$, the computational complexity of the resulting error correction procedure for AMD codes can be drastically reduced for large b or t compared to the method shown in Example 2.1.

Definition 2.3: [10] A Code $C \subset GF(2^n)$ with $|C| > R = \max_{0 \neq e \in GF(2^n)} |\{c | c \in C, c \oplus e \in C\}|$ is called a R-robust error detection code.

Theorem 2.2: Let $V_1 = \{y, x, f(y,x)\}, y \in GF(2^k), x \in GF(2^m), f(y,x) \in GF(2^r)$ be a (k, m, r) AMD code constructed by Theorem 2.1, where $k = sr, m = tr$. Let $V_2 = \{(x, g(x))\}, x \in GF(2^m), g(x) \in GF(2^{r_x})$ be a R-robust error detection code. Suppose

$$f(y,x) = y_1M_1(x) \oplus y_2M_2(x) \oplus \cdots \oplus y_sM_s(x) \oplus A(x), \quad (8)$$

where $y_i \in GF(2^r), 1 \leq i \leq s$ and $A(x), M_i(x), 1 \leq i \leq s$ are monomials of x (see (3) and (5)). The degree of $M_i(x)$ is between 1 and $b+1$. Then the code C composed of all vectors $(y, x, g(x), f(y,x))$ can correct an error $e = (e_y, e_x, e_g, e_f)$ with any multiplicity (number of 1's in the n-bit binary error vector, $n = k + m + r$) if e stays for at least $T = \max\{s+1, R+1\}$ clock cycles and the matrix

$$M = \begin{pmatrix} 1 & M_1(x(1)) & M_2(x(1)) & \cdots & M_s(x(1)) \\ 1 & M_1(x(2)) & M_2(x(2)) & \cdots & M_s(x(2)) \\ & \cdots & \cdots & \cdots & \cdots \\ 1 & M_1(x(T)) & M_2(x(T)) & \cdots & M_s(x(T)) \end{pmatrix}$$

has a rank of $s+1$, where $x(i), 1 \leq i \leq T$ is the random data x at the i^{th} clock cycle.

Remark 2.1: Let $t = 1$ in Theorem 2.1. Then $k = br$ and $s = b$ (see Theorem 2.2). Select $\{(x, g(x))\}$ to be a robust duplication code with $R = 2$, where $g(x) = x^3$ and $r_x = r$. In this case it is easy to verify that M is a Vandermonde matrix

$$M = \begin{pmatrix} 1 & x(1) & x^2(1) & \cdots & x^s(1) \\ 1 & x(2) & x^2(2) & \cdots & x^s(2) \\ & \cdots & \cdots & \cdots & \cdots \\ 1 & x(T) & x^2(T) & \cdots & x^s(T) \end{pmatrix} \quad (9)$$

The rank of the matrix is $s+1 = b+1$ as long as there are at least $s+1$ different $x(i)$ in $\{x(1), x(2), \cdots, x(T)\}$.

Example 2.2: Let $k = m = r_x = r = 3$ $(s = t = 1)$. Let $g(x) = x^3$ and $f(y,x) = y \cdot x \oplus x^3$. Then $V_1 = \{(y, x, f(y,x))\}$ is a AMD code with $Q_{V_1} = 0.25$. V_2 is a 2-robust error detection code and a 3-strong robust error

TABLE I
ORIGINAL, DISTORTED AND PARTIALLY CORRECTED CODEWORDS (ERRORS IN $\tilde{x}$ AND $\tilde{g}(x)$ ARE CORRECTED) IN EXAMPLE 2.2 (ALL VALUES ARE IN $GF(2^3)$). $e = (e_y = 3, e_x = 2, e_g = 7, e_f = 1)$.

Original Codeword			Distorted Codeword				Codeword with Corrected x and $g(x)$			
	y	xg(x)	f(y,x)	$\tilde{y}$	$\tilde{x}$	$\tilde{g}(x)$	$\tilde{f}(y,x)$	$\tilde{y}$	x	$g(x)$	$\tilde{f}(y,x)$
$c(1)$	1	2 3	1	$\tilde{c}(1)$ 2	0	4	0	$\tilde{c}(1)$ 2	2	3	0
$c(2)$	3	5 6	2	$\tilde{c}(2)$ 0	7	1	3	$\tilde{c}(2)$ 0	5	6	3
$c(3)$	4	4 5	3	$\tilde{c}(3)$ 7	6	2	2	$\tilde{c}(3)$ 7	4	5	2

correction code. According to Theorem 2.2, an error can be corrected by $C = \{(y, x, g(x), f(y, x))\}$ if it stays for at least 3 consecutive clock cycles.

Consider the three codewords $c(1), c(2), c(3)$ of the code C shown in the first column of Table I. Suppose $c(1), c(2), c(3)$ are distorted by the same error $e = (e_y = (011) = 3, e_x = (010) = 2, e_g = (111) = 7, e_f = (001) = 1)$, where $e_g = \tilde{x}^3 \oplus x^3$. (The generator polynomial of $GF(2^3)$ is selected to be $\pi(z) = z^3 \oplus z \oplus 1$. All 3-bit binary vectors are represented by their decimal equivalents.) The distorted codewords are shown in the second column of Table I. Following the algebraic decoding algorithm described in [10], e_x and e_g can be successfully corrected and the original x can be retrieved for all the three codewords. The partially corrected codewords after the correction of errors in $\tilde{x}$ and $\tilde{g}(x)$ are shown in the third column of Table I. The three syndromes of V_1 can be computed as follows.

$$S_1 = \tilde{y}x \oplus x^3 \oplus \tilde{f}(y, x) = 2e_y \oplus e_f = 7, (c_1)$$
$$S_2 = \tilde{y}x \oplus x^3 \oplus \tilde{f}(y, x) = 5e_y \oplus e_f = 5, (c_2)$$
$$S_3 = \tilde{y}x \oplus x^3 \oplus \tilde{f}(y, x) = 4e_y \oplus e_f = 6, (c_3)$$

For this example $s = 1, R = 2, T = 3$ and

$$M = \begin{pmatrix} 1 & 2 \\ 1 & 5 \\ 1 & 4 \end{pmatrix} \quad (10)$$

Using any of the two syndromes, we can get $e_y = 3, e_f = 1$. The third syndrome may be used to verify that the error was the same for all three distorted codewords $\tilde{c}_1$, $\tilde{c}_2$ and $\tilde{c}_3$. If the third syndrome is not zero after the error correction, the correction is unsuccessful. The user data should be abandoned and an error flag should be asserted.

Remark 2.2: We note that when V_1 is a (br, r, r) AMD code and V_2 is a 3-robust error correction code composed of all codewords $(x, g(x) = x^3)$, the memory overhead is $3b^{-1}$. Any nonzero error is masked with a probability of at most $(b + 1)2^{-r}$. A system of three linear equations needs to be solved for correcting errors in y and $f(y, x)$.

B. Error Correction Procedure and Properties for AMD Codes

According to Theorem 2.2, the correction of repeating errors by AMD codes described in Theorem 2.2 is completed in $T = \max\{s+1, R+1\}$ clock cycles after detecting a nonzero error. The correction procedure can be split into two steps. In the first step, errors in x and $g(x)$ are corrected using the error correction algorithm for robust codes shown in [10].

TABLE II
AMD ROBUST ERROR CORRECTION CODES $\{(y, x, g(x), f(y, x))\}$ FOR PROTECTING MEMORIES WITH $k = 32$.

b	t	m	r_x	r	Q_V	Clock Cycles Required for Correction	Percentage Overhead $(r + r_x)/k$
3	1	11	11	11	2^{-8}	4	103.125%
5	1	7	7	7	$6 \cdot 2^{-7}$	6	65.625%
7	1	5	5	5	2^{-2}	8	46.875%
1	2	22	22	11	2^{-10}	3	171.875%

If different errors in x are detected in the T consecutive clock cycles, an error flag is asserted and no further error correction is attempted. In the second step, a system of linear equations is solved to correct errors in y and $f(y, x)$. If there are no solutions for the system, the errors are uncorrectable and the error flag will be asserted. Errors can be successfully corrected if and only if the same error stays for at least T clock cycles and there are at least $s + 1$ different $x(i)$ among $\{x(1), x(2), \cdots, x(T)\}$, where $x(i)$ is the m-bit random vector at the i^{th} clock cycle (see Remark 2.1 and [10]).

The laziness [10] of errors can be defined by the following equation.

$$P_R = P(e(\tau + 1) = e(\tau), e(\tau) \neq 0). \quad (11)$$

When $t = 1$ the theoretical error correction rate C_R can be calculated as $C_R = P_R^{T-1} \prod_{i=1}^{T-1}(1 - i \cdot 2^{-m})$, which is the strictly increasing function of P_R and the strictly decreasing function of T. Similar properties of C_R can be observed for $t > 1$. (see Figure 1);

As illustrative examples, we next consider four different alternatives for protecting memories with $k = 32$ information bits(Table II). The first three alternatives have $t = 1$, $g(x) = x^3$, $m = r_x = r$ and $b = 3, 5, 7$ respectively. The fourth alternative has $t = 2$, $b = 1$ and $m = r_x = 2r$. For this code $x, g(x) \in GF(2^{2r})$. Let $x = (x_1, x_2)$, where $x_1, x_2 \in GF(2^r)$. Then $g(x) = (x_1^3, x_2^3)$. $\{x, g(x)\}$ is the concatenation of two independent robust duplication codes.

In Table II, we compare the number of redundant bits, the worst case error masking probability Q_V, the number of clock cycles required for error correction and the percentage overhead for all alternatives. Generally speaking, codes with smaller b and larger r have better worst case error masking probability Q_V. Moreover, these codes require a smaller number of clock cycles for error correction (see rows 2 and 5 in Table II). However, these codes need more redundant bits in the memory and have larger area overhead than other alternatives.

To verify the correctness of C_R, $10,000$ rounds of simulations have been conducted in MATLAB for the four alternatives shown in Table II. During each round of the simulation, the laziness P_R is increased from 0 to 1 by a step of 0.1. For each value of P_R, we randomly select a codeword (not necessarily different) for each clock cycle. The number of clock cycles required for error correction for each alternative is shown in Table II. A nonzero error is randomly generated and inserted during the first clock cycle. For the following clock cycles, the nonzero error has a probability of P_R to repeat. If the nonzero error does not repeat, another error pattern is randomly selected among all the other possible error

patterns (including the all-zero error pattern). The simulated error correction rate is shown in Figure 1. As expected, C_R increases as P_R increases. When $P_R = 1$, C_R is around $1 - 2^{-r}$. Larger r and/or smaller b result in a better error correction rate when P_R is fixed, which is at the cost of larger hardware overhead due to the increased number of redundant bits (see Table II).

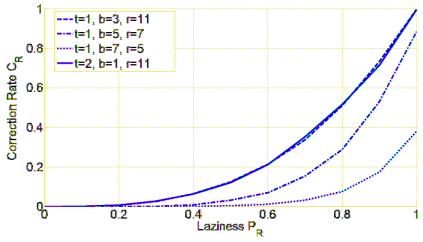

Fig. 1. Error Correction Rate for the Four Alternatives Shown in Table II

There is a small chance of miscorrection by robust codes $\{x, g(x)\}$, which may result in miscorrection for AMD codes as well. The observed miscorrection rate during the simulation is very small ($<< 0.01$).

III. Software and Hardware Co-Design for Error Detection and Correction Using Algebraic Manipulation Correction Codes

A. Complexity for Encoding and Syndrome Computation

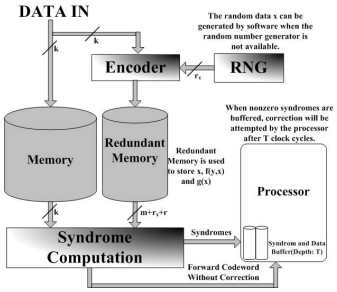

Fig. 2. Reliable and Secure Memory Architecture Protected by Modified AMD Codes

Figure 2 shows the general reliable and secure memory architecture using the robust AMD codes described in Theorem 2.2. Extra memory blocks are needed to store x, $g(x)$ and $f(y, x)$. The area overhead is mainly determined by the size of the extra memory blocks, thus the number of redundant bits $m + r_x + r$ of the code (see Table II). The encoder and the syndrome computation block are implemented in hardware. The error correction algorithm, which involves inverse operation in the Galois field $GF(2^{r_x})$ and solving a system of linear equations, is implemented in software when necessary. During each clock cycle, the syndromes are computed and transferred to the processor together with the possibly distorted codewords (outputs of the original and the redundant memories). The syndromes and the codewords are buffered by the processor. The depth of the buffer is T, which is the number of codewords required for error correction. When a nonzero syndrome is buffered, the processor will wait for another $T - 1$ clock cycles and then try to correct the past T codewords according to the buffered syndromes. If the error cannot be corrected, an error flag signal will be asserted and the past T user data will be abandoned. When no error is detected, the user data will be immediately processed by the processor.

The encoder computing $g(x)$ and $f(y, x)$ requires addition, power operation and multiplication in Galois filed $GF(2^r)$ and $GF(2^{r_x})$. The syndrome computation block contains almost the same operations as the encoder. Depending on applications, different designs for the encoder can be applied. For speed critical applications, the Galois field multiplications and power operations can be paralleled. This method requires larger area overhead but can result in a encoding network with a smaller latency. For area critical applications, encoding based on Horner scheme [11] can be adopted.

The encoders for the four codes shown in Table II are modelled in Verilog and synthesized in RTL Design Compiler using 45nm NANGATE library [12]. The area, latency and the power consumption for designs of encoders based on paralleled implementation of Galois field multiplications and power operations or Horner Scheme are shown in Table III.

TABLE III
Area, Latency and Power for Different Encoder Designs for Codes Shown in Table II

	Parallel Implementation				Horner Scheme			
t	1	1	1	2	1	1	1	2
b	3	5	7	1	3	5	7	1
Area	1853.2	1291.7	794.5	2195.3	1529.5	886.8	584.7	1838.9
Power (μW)	267.47	165.01	95.22	369.61	434.92	248.03	155.90	355.57
Latency (ps)	1437	1216	1216	1511	2712	3045	3764	1618

IV. Conclusions

In this paper we present the error correction algorithm for robust algebraic manipulation detection codes. Using the proposed algorithm, any repeating error (error with a high laziness P_R) can be corrected with a probability converging to one as P_R or the number of redundant bits r grows regardless of the error multiplicity. The area, power consumption and the latency for the encoder and the syndrome computation circuit are studied.

References

[1] Z. Ming, X. L. Yi, L. Chang, and Z. J. Wei, "Reliability of memories protected by multibit error correction codes against mbus," *Nuclear Science, IEEE Transactions on*, vol. 58, no. 1, pp. 289 –295, feb. 2011.

[2] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerers apprentice guide to fault attacks," 2002.

[3] S. Skorobogatov, "Optical fault masking attacks," *Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 23–29, 2010.

[4] M. G. Karpovsky and A. Taubin, "New class of nonlinear systematic error detecting codes," *IEEE Transactions on Information Theory*, vol. 50, no. 8, pp. 1818–1820, 2004.

[5] Z. Wang, M. Karpovsky, and K. Kulikowski, "Design of memories with concurrent error detection and correction by nonlinear SEC-DED codes," *Journal of Electronic Testing*, pp. 1–22, 2010.

[6] Z. Wang, M. Karpovsky, and A. Joshi, "Reliable MLC NAND flash memories based on nonlinear t-error-correcting codes," in *Dependable Systems and Networks, IEEE/IFIP International Conference on*, 2010.

[7] Z. Wang and M. Karpovsky, "Algebraic manipulation detection codes and their applications for design of secure cryptographic devices," in *On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International*, july 2011, pp. 234 –239.

[8] ——, "New error detecting codes for the design of hardware resistant to strong fault injection attacks," in *International Conference on Security and Management*, 2012.

[9] R. Cramer, Y. Dodis, S. Fehr, C. Padr, and D. Wichs, "Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors," in *Advances in Cryptology C EUROCRYPT 2008*, 2008, vol. 4965, pp. 471–488.

[10] K. Kulikowski and M. Karpovsky, "Robust correction of repeating errors by non-linear codes," *Communications, IET*, vol. 5, no. 16, pp. 2317 – 2327, 4 2011.

[11] M. Ceberio and V. Kreinovich, "Greedy algorithms for optimizing multivariate Horner schemes," *SIGSAM Bull.*, vol. 38, pp. 8–15, March 2004. [Online]. Available: http://doi.acm.org/10.1145/980175.980179

[12] "Nangate 45nm open cell library," `http://www.nangate.com`.