Secure Memories Resistant to Both Random Errors and Fault Injection Attacks Using Nonlinear Error Correction Codes

Shizun Ge, Zhen Wang, Pei Luo, Mark Karpovsky*
Reliable Computing Laboratory, Boston University, Boston , USA
shizunge@bu.edu, wang.zhen.mtk@gmail.com, luopei@bu.edu, markkar@bu.edu

***Abstract*—Memories used in cryptographic devices are vulnerable to fault injection attacks. To mitigate the danger of these attacks, error control codes are often used in memories to detect maliciously injected faults. Most of codes proposed for memories in cryptographic devices are error detecting codes with small Hamming distances that cannot be for error correction. While being able to provide sufficient protection against fault injection attacks, these codes cannot provide a satisfactory reliability under the presence of random errors. In this paper we present reliable and secure memory architectures based on two nonlinear error correcting codes. The presented coding technique can be used for detection of fault injection attacks as well as for correction of random errors. The construction and the error correction procedures for the code will be described. The error handling methodology used to distinguish between random errors and maliciously injected faults will be discussed.**

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. SRAMs, for example, are often used as caches and internal memories in embedded systems. Non-volatile memories like EEPROM and Flashes are often used in cryptographic devices to store secret informations such as the encryption keys and passwords.

In secure applications, the security of memories and the whole system is threatened by side-channel attacks such as fault injection attacks [2], [3]. Providing satisfactory security to memories is becoming more and more challenging and important in modern System-On-Chip (SOC) designs thanks to the ubiquitous usage of cryptography algorithm in digital devices.

Robust codes and their variants based on **nonlinear encoding functions** were proposed in [8], [9], [10] to protect cryptographic devices against fault injection attacks. Compared to linear codes based on linear encoding functions such as parity codes, Hamming codes and BCH codes, nonliear robust codes can provide nearly equal protection against all error patterns and are more suitable for detecting malicious injected faults, where the error model is impossible to predict due to the adaptive nature of the attacker.

One limitation of robust codes is that these codes assume the information bits of messages or outputs of the device-to-be-protected are uniformly distributed and are not controllable by external forces, e.g. by an attacker during error injection attacks on devices. The reliability and the security of the communication or computation channels protected by robust codes will be largely compromised if both information bits of the messages and the non-zero error patterns can be controlled by the attacker.

To overcome the limitation of robust codes, algebraic manipulation detection (AMD) codes are presented in [12], [13] for the protection of cryptographic devices against the strongest attackers. However, the constructions presented in [12], [13] usually generate codes with a Hamming distance of 1, which cannot be used for error correction. While the resulting AMD codes are suitable for protecting the secure devices against fault injection attacks, in certain circumstances they may not be able to provide enough resistance against random transient errors introduced by the mother nature. In this paper, we discuss two modifications of AMD codes. These two modifications can effectively increase the distance of AMD codes so that they can be used for not only detecting maliciously injected faults but also for correcting single-bit random errors caused by natural reasons. The proposed codes can provide a guaranteed level of security as well as a high level of reliability to the protected device.

The rest part of the paper is organized as follows. In Section II, previous works on AMD codes are briefly introduced. The definition of Algebraic Manipulation Correction (AMC) codes is described. In Section III, we described the construction of nonlinear single-error-correcting codes based on concatenation. In Section IV, the construction of algebraic manipulation correction codes the error correction algorithm are described. The error handling mechanism used to distinguish between random errors and fault attacks are discussed in Section V.

II. PREVIOUS WORK

A. AMD Codes

AMD codes are designed to provide a guaranteed level of security even if the attacker can control both the error patterns and the input (thus the fault-free output) of a device. Different from regular error control codes, a codeword of an AMD code contains three parts: k-bit user defined information y, m-bit random data x and r-bit redundancy $f(y,x)$.

Throughout the paper we denote by $\oplus$ the addition in $GF(q), q = 2^r$. All the results presented in the paper can be easily generalized to the case where $q = p^r$ (p is a prime). An AMD code V with codewords $(y, x, f(y, x))$, where $y \in GF(2^k), x \in GF(2^m)$ and $f(y,x) \in GF(2^r)$, will be referred to as a (k, m, r) code.

Definition 2.1: (**Security Kernel**) [12] For any (k, m, r) error detecting code V with the encoding function $f(y, x)$, where $y \in GF(2^k), x \in GF(2^m)$ and $f(y, x) \in GF(2^r)$, the **security kernel** K_S is the set of errors $e = (e_y, e_x, e_f), e_y \in GF(2^k), e_x \in GF(2^m), e_f \in GF(2^r)$, for which there exists y such that $f(y \oplus e_y, x \oplus e_x) \oplus f(y, x) = e_f$ is satisfied for all x.

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

For cryptographic devices and secure applications, non-zero errors e in the security kernel can be used by an advanced attacker to bypass the protection based on the error detecting code V. For any error $e^* = (e^*_y, e^*_x, e^*_f) \in K_S, e^* \neq 0$, there exists y^* (the protected information at the output of the device) such that for this y^* the error e^* is not detected for any choice of the random variable x (the probability of not detecting e^* for the information y^* is equal to 1). Thus to conduct a successful attack, it is sufficient for the attacker to inject $e^* \in K_S$ when the expected output is in the format of $(y^*, x, f(y^*, x))$. An AMD code should have no errors in the security kernel except for the all zero vector in $GF(2^n)$, where $n = k+m+r$.

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

Definition 2.2: [14] 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 $GF(2^n)$, $n = k + m + r$.

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

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

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

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

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

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

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

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

Theorem 2.1: [12] 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 $Q_V = \max_y \max_{e \neq 0} Q_V(y, e) = (b + 1)2^{-r}$.

Remark 2.1: The construction and properties of AMD codes shown in Theorem 2.1 are tightly related to the q-ary Generalized Reed-Muller codes.[12]

A special case of Theorem 2.1 is $t = 1, b < 2^r - 1$, b is odd. In this case $f(y, x) = \oplus_{i=1}^{b} y_i x^i + x^{b+2}$.

Example 2.1: Let $t = b = 1$, then $m = r$ and $k = (\binom{t+b+1}{t} - 1 - t)r = r$. The encoding function $f(y, x)$ for the AMD code based on Theorem 2.1 is $f(y, x) = y \cdot x \oplus x^3$, where $x, y, f(y, x) \in GF(2^r)$. The resulting AMD code has $Q_V = 2^{-r+1}$.

If $t = 1$ and $b = 3$, then $m = r$, $k = (\binom{t+b+1}{t} - 1 - t)r = 3r$ and $f(y, x) = y_1 \cdot x \oplus y_2 \cdot x^2 \oplus y_3 \cdot x^3 \oplus x^5$, where $y_1, y_2, y_3, x, f(y, x) \in GF(2^r)$. For this code $Q_V = 2^{-r+2}$.

Generally speaking, AMD codes constructed in Theorem 2.1 have a small Hamming distance and cannot be directly used for error correction.

Definition 2.3: An AMD code whose Hamming distance is at least 3 is called an Algebraic Manipulation Correction (AMC) code.

For secure applications, AMD codes described in the Section II-A can provide guaranteed level of security under the strongest attacker model by detecting the injected faults with a high probability. We assume the attacker knows every detail of the device including the error control code used to protect the device. The attacker can select specific inputs to the device during fault injection attacks. (The attacker can thereby control the fault-free outputs). For the case of memories, the attacker may have information or even precise knowledge of the data stored in a number of different memory locations. Moreover, the attacker is also able to inject any specific error pattern to the targeted memory location. Under such an advanced attacker model, only AMD codes can provide sufficient protection against fault injection attacks [12], [13]. To our best knowledge, all the known fault injection mechanisms can only provide a limited timing resolution. For instance, the time between two consecutive shot of the laser gun, which is one of the most powerful fault injection methods, is affected by the speed of recharging and the delay between the trigger signal and the shot [15]. Thereby, the same injected faults are very likely to stay for several consecutive clock cycles and cause repeating errors at the output of the secure memories, in which case the AMD codes not only detect the faults but also correct the errors. To distinguish between errors which appear due to natural causes and malicious attacks, a counter and an adaptive threshold can be set up so that once a certain number of uncorrectable errors occur to the secure memories, the device is disabled.

The problem of distinguishing between malicious and random errors due to natural causes will be considered in Section V

III. Single-Error-correcting AMC Codes Based on Concatenations

A straightforward method for increasing the distance of AMD codes is to concatenate it with a linear error correcting codes. In this Section, we discuss the error correcting procedure for codes based on concatenating AMD codes with Hamming codes.

Theorem 3.1: Let code $V_{AMD} = \{(y, x, f(y, x))\}$ be an AMD code defined in Theorem 2.1 with parameters $m = r$ and $k = bm$ and the nonlinear encoding function $f(y, x) = y_1 x \oplus y_2 x^2 \oplus y_3 x^3 \oplus \cdots \oplus y_b x^b \oplus x^{b+2}$, where $y \in GF(2^{bm})$ is the information part, $x \in GF(2^m)$ is the random part and $f(y, x) \in GF(2^m)$ is the redundant part. (b is odd and $b < 2^m - 1$. , $y = (y_1, y_2, \ldots, y_b) \in GF(2^k)$, $y_i \in GF(2^m)$ for $i \in \{1, 2, \ldots, b\}$)

Let $V_H = \{(v_h, v_h P)\}$ be the Hamming code with distance 3, where $v_h \in GF(2^{(b+2)m})$ is the information part, $v_h P \in GF(2^{r_V})$ is the redundant part, $r_{V_H} = \lceil log_2((b + 2)m + r_{V_H} + 1)\rceil$. Let $v_h = (y, x, f(y, x)) \in V_{AMD}$.

Then the code $V_{con} = \{(y, x, f(y, x), v_h P)\}$ is a SEC AMC code where any error is masked by a probability of at most $Q_{V_{con}} = (b + 1)2^{-m}$ (b is odd). The distance for this code is 3.

Proof: This code obviously can correct single error.

Let error vector be $e = (e_1, e_2, e_3, e_4)$, where $e_1 \in GF(2^{bm})$, $e_2 \in GF(2^m)$, $e_3 \in GF(2^m)$, $e_4 \in GF(2^{r_{V_H}})$. Let $e_{v_h} = (e_1, e_2, e_3) \in GF(2^{(b+2)m})$. Also let $e_1 = (e_{11}, e_{12}, \ldots, e_{1b})$, where $e_{1i} \in GF(2^m)$ for $i \in \{1, 2, \ldots, b\}$. The error masking equations can be written as

$$\begin{gathered} f(y \oplus e_1, x \oplus e_2) = f(y, x) \oplus e_3, \\ P(v_h \oplus e_{v_h}) = v_h P \oplus e_4. \end{gathered}$$

Expand above equations, we have

$$\begin{aligned} \mathbf{0} &= (y_1 \oplus e_{11})(x \oplus e_2) \oplus \cdots \oplus (y_b \oplus e_{1b})(x \oplus e_2)^b \\ &\quad \oplus x^{b+2} \oplus y_1 x \oplus \cdots \oplus y_b x^b \oplus x^{b+2} \oplus e_3, \end{aligned} \tag{6}$$

$$\mathbf{0} = P e_{v_h} \oplus e_4. \tag{7}$$

1) If $(e_1, e_2, e_3, e_4) = \mathbf{0}$, the error will not affect the messages.
2) If $e_{v_h} = (e_1, e_2, e_3) = \mathbf{0}$ and $e_4 \neq \mathbf{0}$, the Equation (7) will not be satisfied. The error will always be detected.
3) If $(e_1, e_2) = \mathbf{0}$ and $e_3 \neq \mathbf{0}$, the Equation (6) will not be satisfied. The error will always be detected.
4) If $(e_1, e_2) \neq \mathbf{0}$, there are at most $b + 1$ solutions of x for Equation (6). Thus any error will be masked with a probability of at most $Q_{V_{con}} = (b + 1)2^{-m}$, where 2^m is the number of possible values of x.

■

A generalized proof for the probability for the detection for concatenations of nonlinear code and linear code can be found in [16].

Remark 3.1: We may use any AMD encoding functions which are describe in Theorem 2.1. The distance and error masking probabiltiy will not be affected.

Example 3.1: There is an AMD code $V_{AMD}\{(y, x, f(y, x))\}$ with parameter the number of information bits $k = br = 35$, the random bits $m = 7$, the redundant bits $r = 7$ and the degree of nonlinear encoding function $b = 5$. The nonlinear encoding function is $f(y, x) = y_1x \oplus y_2x^2 \oplus \dots y_5x^5 \oplus x^7$, where $y = (y1, y2, \dots, y_5) \in GF(2^{35})$ are the information part, $y_i \in GF(2^7)$ for $i = 1, 2, \dots, 5$, $x \in GF(2^7)$ are the random number.

Let $v_h \in V_{AMD}$ be a codeword of the above AMD code. By adding $v_hP \in GF(2^7)$, which contains 6 more redundant bits, to the end of the AMD code, we form a SEC code $V_{con} = \{(y, x, f(y, x), v_hP)\}$ based on concatenations, where P is the encoding matrix for the $(55, 49, 3)$ Hamming code.

Assume there is an error vector $e = (e_1, e_2, e_3, e_4)$ corrupts the original messages $v = (v_1, v_2, v_3, v_4)$ of V_{con}, where $v_1 = y$, $v_2 = x$, $v_3 = f(y, x)$, and $v_4 = v_hP$. Denote the distorted message by $\tilde{v} = (\tilde{v_1}, \tilde{v_2}, \tilde{v_3}, \tilde{v_4})$, where $\tilde{v_1} = v_1 \oplus v_1$, $\tilde{v_2} = v_2 \oplus v_2$, $\tilde{v_3} = v_3 \oplus v_3$, and $\tilde{v_4} = v_4 \oplus v_4$. e_1, v_1, and $\tilde{v_1} \in GF(2^{bm})$. e_2, v_2, and $\tilde{v_2} \in GF(2^m)$. e_3, v_3, and $\tilde{v_3} \in GF(2^m)$. e_4, v_4, and $\tilde{v_4} \in GF(2^{r_{V_H}})$.

The correction algorithm for code V_{con} is

1) Calculate the syndrome of V_H, and find the corresponding error $\hat{e} \in GF(2^{(b+2)m+r_V})$. $||\hat{e}|| = 1$. If $\hat{e}$ cannot be found, then the error e is not correctable by the code; no further steps will run.
2) Calculate $\hat{v} = \tilde{v} \oplus \hat{e} = (\hat{v_1}, \hat{v_2}, \hat{v_3}, \hat{v_4})$, where $\tilde{v_1} \in GF(2^{bm})$, $\tilde{v_2} \in GF(2^m)$, $\tilde{v_3} \in GF(2^m)$, $\tilde{v_4} \in GF(2^{r_{V_H}})$ and $\hat{v} \in GF(2^{(b+2)m+r_{V_H}})$.
3) Calculate the syndrome $S_{con} = f(\hat{v_1}, \hat{v_2}) \oplus \hat{v_3}$. If $S_{con} = \mathbf{0}$, then the single error is corrected; the corrected message is $\hat{v}$. Otherwise the error e is not correctable for the code.

Theorem 3.2: The concatenation codes V_{con} constructed in Theorem 3.1 have no errors miscorrected by all codewords. Any nonzero error will be miscorrected with a probability of at most $Q_{mc} = (b + 1)2^{-m}$. (The probability for the miscorrection is defined as $Q_{mc}(v, e) = |\{x|v \neq \hat{v}\}|2^{-m}$, where 2^m is the number of possible values of the random number x.)

Proof: In the correction algorithm, an error e will be miscorrected if and only if there exist $\hat{e}$, $\hat{e} \neq e$ $S_{con} = \mathbf{0}$.

Let $\hat{e} = (\hat{e_1}, \hat{e_2}, \hat{e_3}, \hat{e_4})$, where $\hat{e_1} \in GF(2^{bm})$, $\hat{e_2} \in GF(2^m)$, $\hat{e_3} \in GF(2^m)$, $\hat{e_4} \in GF(2^{r_V})$. There is only one term among $\hat{e_1}, \hat{e_2}, \hat{e_3}, \hat{e_4}$ is not $\mathbf{0}$, since $||\hat{e}|| = 1$.

Rewrite the equation $S_{con} = f(\hat{v_1}, \hat{v_2}) \oplus \hat{v_3} = \mathbf{0}$ as

$$S_{con} = f(v_1 \oplus e_1 \oplus \hat{e_1}, v_2 \oplus e_2 \oplus \hat{e_2}) \oplus f(v_1, v_2) \oplus e_3 \oplus \hat{e_3} = \mathbf{0}. \tag{8}$$

Let H be the parity check matrix of V. When $v = (v_1, v_2, v_3, v_4)$ is miscorrected as $\hat{e} = (\hat{e_1}, \hat{e_2}, \hat{e_3}, \hat{e_4})$, we have $He = H\hat{e}$. Thereby $e \oplus \hat{e}$ is a codeword of V. If $e \neq \hat{e}$, then $(e_1, e_2, e_3) \neq (\hat{e_1}, \hat{e_2}, \hat{e_3})$. Otherwise to guarantee that $e \oplus \hat{e}$ is a codeword of V, e_4 has to equal to $\hat{e_4}$, which contradicts to the assumption that $e \neq \hat{e}$. Thereby $\hat{e}$ can be divided into following cases.

1) If $(e_1, e_2) \neq (\hat{e_1}, \hat{e_2})$. From [17][16] we know there are at most $b + 1$ solutions to the Equation (8). Meanwhile there are 2^m possible values of x. Thereby, the error will be miscorrected with a probability of at most $Q_{mc} = (b + 1)2^{-m}$.
2) If $(e_1, e_2) = (\hat{e_1}, \hat{e_2})$ and $\hat{e_3} \neq \mathbf{0}$. The error will always be detected since Equation (8) will never be satisfied.

■

Similar to the original AMD codes in [17], the AMC codes based on concatenations also has no undetectable errors. And the error masking probabilities are same for both codes if they use the same nonlinear encoding function. Additionally, the codes based on concatenations can correct single bit errors, and no errors are miscorrected by all codewords. These features enable the codes V_{con} to protect memories for strong attack model as well as to prevent memories from random errors. At a cost, additional r_{V_H} bits should be added to the original AMD codes.

Figure 2 presents the architecture for memories protected by the AMC code based on concatenations.

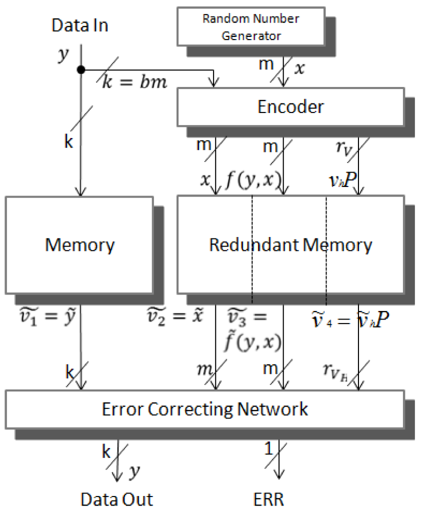

Fig. 1. Memory protected by the AMC code based on concatenations

IV. Single-Error-Correcting AMC Codes

In this Section, we will present the general construction of Algebraic Manipulation Correction codes. These codes have the same error correcting capability while requiring less redundant bits compared to the straightforward construction based on concatenation described in the last Section.

Theorem 4.1: Suppose

1) (x, xP) is a codeword of $(m + r_H, m, 3)$ binary linear Hamming code V_H with r_H redundant bits and distance 3, where $x \in GF(2^m)$, $xP \in GF(2^{r_H})$, P is a $r_H \times m$ encoding matrix, and
2) $f(y, x) \in GF(2^m)$ is a nonlinear encoding function $f(y, x) = y_1x \oplus y_2x^2 \oplus y_3x^3 \oplus \cdots \oplus y_bx^b \oplus x^{b+2}$ (b is odd, $b+2 < 2^m - 1$). where $y = (y_1, y_2, \dots, y_b)$; $y_i \in GF(2^m), (i = 1, 2, \dots, b)$; $x \in GF(2^m)$; $f(y, x) \in GF(2^m)$ and all the operations are in $GF(2^m)$; $2^m - 1$ is a prime number;
3) $\pi y = y_1 \oplus y_2 \oplus y_3 \oplus \cdots \oplus y_b \in GF(2^m)$ is the byte-wise parity of y;

Then the code $V_{AMC} = \{(y, \pi y \oplus x, xP, f(y, x))\}$ is a $(k, m, m + r_H)$ SEC *Algebraic Manipulation Correction* (AMC) code, with $k = bm$ information bits, m random bits, $m + r_H$ redundant bits.

This code has secure kernel $K_{V_{AMC}} = \{\mathbf{0}\}$ with the maximum error masking probability $Q_{V_{AMC}} = (b + 1)(2^m - 2)^{-1}$ and Hamming distance 3.

Remark 4.1: We may use any AMD encoding functions $f(y, x)$ described in [17], [12], [13]. In general case, $y \in GF(2^k)$, where $k = sr$, and $x \in GF(2^m)$, where $m = tr$. In this case, $x = (x_1, x_2, \dots, x_t)$ and $f(y, x)$ is a polynomial of t variables

$x_1, \ldots, x_t$. Thus, y will be divided into s/t parts, each of which contains tr bits, and then $\pi y \in GF(2^{tr})$ is the byte-wise parity. The padding zeros may be applied to y, when s is not dividable by t.

Example 4.1: Let $m = 7$ which is the number of random bits. Also let the encoding function be $f(y,x) = y_1x \oplus y_2x^2 \oplus y_3x^3 \oplus y_4x^4 \oplus y_5x^5 \oplus x^7$, where $y = (y_1, y_2, \ldots, y_5) \in GF(2^{35})$ is the information part, $y_i \in GF(2^7)$ for $i = 1, 2, \ldots, 5$, $x \in GF(2^7)$ is the random number.

Let $\{(x, xP)\}$ be the $(11, 7, 3)$ Hamming code, where P is the encoding matrix for the Hamming code. Since $2^7 - 1$ is a prime number, the code V_{AMC} defined by Theorem 4.1 is an AMC code with $d = 3$ and $Q_{V_{AMC}} = \frac{6}{2^7-2} = \frac{1}{21}$.

Comparing to the AMD Code with $k = br = 35$ $m = r = 7$ and nonlinear encoding function $f(y,x) = y_1x \oplus y_2x^2 \oplus y_3x^3 \oplus y_4x^4 \oplus y_5x^5 \oplus x^7$, the codeword of V_{AMC} contains 4 more redundant bits.

Remark 4.2: In a normal base Galois field [17], square operation can be achieved by the cyclic shift. As a result, $f(y,x)$ in Theorem 4.1 can be slightly modified to reduce its hardware complicity of computing $f(y,x)$ using the following encoding equation

$$f(y,x) = y_1x \oplus y_2x^2 \oplus y_3x^4 \oplus \cdots \oplus y_bx^{2^{(b-1)}} \oplus x^{2^b+1},$$

where $y = (y_1, y_2, y_3, \ldots, y_b)$ and $y_i \in GF(2^m)$ $(i = 1, 2, 3, \ldots b)$; $x \in GF(2^m)$; $x \neq \mathbf{0}, \mathbf{1}$, where $\mathbf{1}$ is all 1 vector; $f(y,x) \in GF(2^m)$; and $2^m - 1$ is a prime number and $b < m$.

This code reduces the computational complexity of decoding at the cost of higher error masking probability which is going up to $Q_{V_{AMC}} = 2^b(2^m - 2)^{-1}$.

A. Algorithm for Single Error Correction and Estimation of Probabilities of Miscorrection for the Proposed Codes

A direct approach is to add codewords to an existing AMD code some additional redundant bits to provide for error correction, $(y, x, f(y,x), P)$ as an example. We will present another approach which can detect and correct the errors in the codewords but will requires less redundant bits.

1) Error correction algorithm for the proposed SEC-DED AMC code: There are four parts in every codeword of the AMC code constructed as in Theorem 4.1, namely y, $\pi y \oplus x$, xP, and $f(y,x)$. For a codeword $v = (v_1, v_2, v_3, v_4)$ of the AMC code V_{AMC} constructed in Theorem 4.1, there are

$v_1 = y = (y_1, y_2, y_3, \ldots, y_b)$; $y_i \in GF(2^m)$, $i = 1, 2, 3, \ldots, b$;
$v_2 = \pi y \oplus x$; $\pi y,\ x,\ v_2 \in GF(2^m)$;
$v_3 = xP$; $xP \in GF(2^{r_H})$;
$v_4 = f(y,x)$; $v_4 \in GF(2^m)$;

(x, xP) is a codeword of a linear Hamming code with distance 3 and the check matrix is $H = [P^T|I]$, where P^T is the transposed matrix of P and I is an identity matrix.

Denote the error vector by $e = (e_1, e_2, e_3, e_4)$ and the received message by $\tilde{v} = (\tilde{v_1}, \tilde{v_2}, \tilde{v_3}, \tilde{v_4})$, where $\tilde{v_i} = v_i \oplus e_i, i = 1, 2, 3, 4$ and $e_1, \tilde{v_1} \in GF(2^{bm})$; $e_2, \tilde{v_2} \in GF(2^m)$; $e_3, \tilde{v_3} \in GF(2^{r_H})$; $e_4, \tilde{v_4} \in GF(2^m)$. We assume that only errors in the information part $v_1 = y$ need to be corrected. The decoding procedure can be divided into the following steps.

1) Calculate $(\tilde{u}, \tilde{v_3})$, where $\tilde{u} = \pi\tilde{v_1} \oplus \tilde{v_2}$
2) Calculate $S_H = H(\tilde{u}, \tilde{v_3})^T$, the syndrome for the Hamming code.
   Use S_H as the input to the Hamming decoder, then obtain the error locator ε, where $\varepsilon \in GF(2^m)$. Since ε is the output of the Hamming decoder, there should be only one bit in ε which is equal to one, and all other bits are zeros.
   Let $u = \tilde{u} \oplus \varepsilon = \pi\tilde{v_1} \oplus \tilde{v_2} \oplus \varepsilon$, where $u \in GF(2^m)$. If uncorrectable multi-bit errors are detected by the Hamming decoder, then no further steps need to be performed. Otherwise, go to the step 3.
3) Calculate S_{AMD} as follows
$$\begin{aligned} S_{AMD} &= f(\tilde{y}, u) \oplus \tilde{v_4} \\ &= f(y \oplus e_1, x \oplus \pi e_1 \oplus e_2 \oplus \varepsilon) \oplus f(y,x) \oplus e_4. \quad (9) \end{aligned}$$
   If both $S_H = \mathbf{0}$ and $S_{AMD} = \mathbf{0}$, then there are no errors. If only $S_{AMD} = \mathbf{0}$, there are multiple errors. Therefore, as long as $S_{AMD} = \mathbf{0}$, the correction procedure is completed. Otherwise go to the next step.
4) Compare S_{AMD} with εu^j, for all $j = 1, 2, 3, \ldots, b$.
   a) If $S_{AMD} = \varepsilon u^j$ for some $j \in \{1, 2, 3, \ldots, b\}$, then the j^{th} part $y_j \in GF(2^m)$ of information $\tilde{v_1} \in GF(2^{bm})$ of the codeword is distorted and the error in that part is $\varepsilon \in GF(2^m)$, which means $\hat{y_j} = \tilde{y_j} \oplus \varepsilon$, where $\hat{y_j} \in GF(2^m)$ is the corrected message.
   b) Otherwise, there are multiple errors or the error is not in the information part v_1. No error correction will be attempted.

The decision table for the proposed single error correction algorithm is summarized in Table I

TABLE I
CORRECTION ALGORITHM DECISION TABLE FOR SEC-DED AMC

S_H	S_{AMD}	Decision
$S_H = \mathbf{0}$	$S_{AMD} = \mathbf{0}$	No Error
	$S_{AMD} \neq \mathbf{0}$	Double/Multiple Errors
$S_H \neq \mathbf{0}$ and $S_H \neq {h_i}^1 (\forall i)$	$\forall S_{AMD}$	Double/Multiple Errors
$S_H = h_i$	$S_{AMD} \neq \varepsilon u^j$ or $S_{AMD} = \mathbf{0}$	Single Error in v_2, v_3, or v_4 Or Double/Multiple Errors
	$S_{AMD} = \varepsilon u^j$ $S_{AMD} \neq \mathbf{0}$	Single Error in v_1 (Correction)

[1] h_i $(1 \leq i \leq m)$ is the i^{th} column of the parity check matrix H of the Hamming code. This row is only valid for non-perfect Hamming code.

Example 4.2: (Single Error Correction)
Consider a proposed AMC code with $b = 2$, $m = 3$. V_H is a $(6, 3, 3)$ Hamming code with $P = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$.

The encoding function is $f(y,x) = y_1x \oplus y_2x^2 \oplus x^5$. The codeword is in the format of $v = ((y_1, y_2), \pi y \oplus x, xP, f(y,x))$, where $y_1,\ y_2,\ x,\ xP, f(y,x) \in GF(2^3)$. We select $z^3 \oplus z \oplus 1$ as the generating polynomial for $GF(2^3)$, with the rightmost bit being the least significant bit.

Suppose $y_1 = (001)$, $y_2 = (001)$, $x = (010)$. Then we have $\pi y \oplus x = (001) \oplus (001) \oplus (010) = (010)$, $(xP)^T = (101)$, and $f(y,x) = (001)(010) \oplus (001)(010)^2 \oplus (010)^5 = (010) \oplus (100) \oplus (111) = (001)$. Thus, the original codeword is $v = (v_1, v_2, v_3, v_4) = (001001, 010, 101, 001)$.

Suppose there is a single error $e = (000010, 000, 000, 000)$ in the received message. Therefore, the distorted message is $\tilde{v} = (001011, 010, 101, 001)$. We have $(\tilde{u}, \tilde{v_3}) = (\pi\tilde{v_1} \oplus \tilde{v_2}, \tilde{v_3}) = (000, 101)$.

$S_H = H(\tilde{u}, \tilde{v_3})^T = [P^T|I](\tilde{u}, \tilde{v_3})^T = (101)$. After decoding $(\tilde{u}, \tilde{v_3})$ using the Hamming decoder, we have $\varepsilon = (010)$. And $u = \pi\tilde{v_1} \oplus \tilde{u} \oplus \varepsilon = (000) \oplus (000) \oplus (010) = (010)$. Then syndrome $S_{AMD} = (001)(010) \oplus (011)(010)^2 \oplus (010)^5 = (011)$. Since $S_{AMD} = \varepsilon u^2 = (010)(010)^2 = 011$, the error $\varepsilon = (010)$ is located at second bit of $\tilde{y_2}$.

The error is successfully corrected.

2) Estimations on a probability for the miscorrection: Suppose $v = (v_1, v_2, v_3, v_4)$, where $v_1 = y$, $v_2 = \pi y \oplus x$, $v_3 = xP$, $v_4 = f(y,x)$ is a codeword for an AMC code V_{AMC} described in Theorem 4.1. Let $e = (e_1, e_2, e_3, e_4)$ be the error vector and $\tilde{v} = \{\tilde{v_1}, \tilde{v_2}, \tilde{v_3}, \tilde{v_4}\}$ be the received (distorted) message, where $\tilde{v_i} = v_i \oplus e_i, i = 1,2,3,4$. Let $e_1 = (e_{11}, e_{12}, \ldots, e_{1b})$, where $e_{1i} \in GF(2^m)$ for $i \in \{1,2,\ldots,b\}$. Denote the message after correction, i.e. the output of the decoder by $\hat{v} = (\hat{v_1}, \hat{v_2}, \hat{v_3}, \hat{v_4})$, where $e_1, \tilde{v_1}, \hat{v_1} \in GF(2^{bm})$, e_2, $\tilde{v_2}$, $\hat{v_2} \in GF(2^m)$; e_3, $\tilde{v_3}$, $\hat{v_3} \in GF(2^{r_H})$; e_4, $\tilde{v_4}$, $\hat{v_4} \in GF(2^m)$.

We say that the error is miscorrected if $c_1 \neq \hat{c_1}$. The miscorrection probability can be defined as

$$Q_{mc}(y,e) = |\{x | v_1 \neq \hat{v_1}, e \neq 0\}| 2^{-m}, \tag{10}$$

where 2^m is the number of possible values of x.

Theorem 4.2: Miscorrection Probability.

For the AMC code constructed by Theorem 4.1, the algorithm in Section IV-A1 has a miscorrection probability $Q_{mc}(y,e)$, at most $b(b+1)(2^m-2)^{-1}$, $\max_{y;e\neq 0} Q_{mc}(y,e) \leq b(b+1)(2^m-2)^{-1}$.

Proof: Consider the following error masking equation:

$$\begin{aligned}\mathbf{0} &= S_{AMD} \oplus \varepsilon u^j \\ &= f(y \oplus e_1, x \oplus \pi e_1 \oplus e_2 \oplus \varepsilon) \oplus f(y,x) \\ &\oplus e_4 \oplus \varepsilon(x \oplus \pi e_1 \oplus e_2 \oplus \varepsilon)^j, \\ &\quad for\ j \in \{1,2,\ldots,b\}.\end{aligned} \tag{11}$$

If Equation (11) holds for some $j \in \{1,2,\ldots,b\}$ and $\|e = (e_1,e_2,e_3,e_4)\| > 1$, then the error will still be treated as a single error ε in y_j, the j^{th} part of information. Thus, the error is miscorrected.

1) If $\pi e_1 \oplus e_2 \oplus \varepsilon \neq \mathbf{0}$, Equation (11) will be an equation with a degree $b+1$, which has at most $b+1$ solutions for x.
2) If $\pi e_1 \oplus e_2 \oplus \varepsilon = \mathbf{0}$, the only situation that the equation always holds is when there is a single error in the information part v_1. Otherwise, Equation (11) will be an equation with a degree smaller than b, which has less than b solutions

We may choose j from $\{1,2,\ldots,b\}$, leading to b equations for x for a given error vector $e = (e_1,e_2,e_3,e_4)$ and information y. For the strong attack model, if the attacker selected e and y carefully, each of these equations will have $b+1$ different solutions in the worst case. Therefore the total number of solutions for x for all the equations is at most $b(b+1)$, while the total number of possible values of x is $2^m - 2$. Thereby, the error miscorrection probability Q_{mc} for a given pair e and y will be $Q_{mc} \leq (b(b+1))(2^m-2)^{-1}$. ■

The AMC code for Theorem 4.1 can be extended to be a code with Hamming distance 4 by adding one more overall parity bit after which the code can correct single error and at the same time detect all double errors without miscorrection of double errors. The error detection and correction capabilities for the extended SEC-DED AMC code is summarized in Table II.

Remark 4.3: We note that the straightforward concatenation approach for contruction of AMC codes with distance 3 based on adding redundant bits to AMD code requires more redundancy than codes constructed by Theorem 4.1.

V. Error Handling

In this section, we describe how to identify attacks and how to distinguish between random errors and attacks.

For a strong error injection attack, we assume that the attackers are able to control the outputs of the protected device, and to apply any non-zero error vector at the outputs to flip the output bits. However due to the limitation of the time precision of the attack methods [17],[2], the errors injected may last for at least A consecutive clock cycles for some $A \geq 1$, which is known to a designer of the system.

Let $\tilde{v}(t)$ and $v(t)$ be the corrupted as a result of the attack and error-free outputs receptively and $\tilde{v}(t) = v(t) \oplus e(t)$. If $e(t) \neq 0$, then $e(t+i) \neq 0$ for $i \in \{1,2,\ldots,A-1\}$. We note that it may happen that $e(t+i) \neq e(t)$.

In addition, we assume that the attacker is able to select specific inputs, thus the outputs, to the device during error injection attacks.

In order to detect the attack, a threshold $T \leq A$ should be calculated. If there are at least T (not necessary consecutive) uncorrectable errors appearing in A consecutive clock cycles, we say there is an attack and disable the device. We use sliding windows of size A to continuously monitor whether there is an attack or not. The upper bound for T is determined by the given targeted probability for the detection of an attack P_{A0}. We would like to achieve $P_{A0} \leq P_A$, where P_A is the probability that T out of A consecutive injections are detected. The lower bound for T is determined by the given targeted probability for the false alarm P_{FA0}. We would like to achieve $P_{FA0} \geq P_{FA}$, where P_{FA} is the probability that random errors occur and are detected in T out A consecutive clock cycles, which will be mistakenly treated as an attack.

The *upper bound* for T will be calculated in the following way, for a given targeted probability for the detection of an attack P_{A0},

Firstly, we know the proposed code with distance 4 can correct all single errors in the information part, detect all double errors and all single errors that are not in the information part, and detect multiple (excluding double) errors with a probability at least $1 - Q_{mc}$. Injection of single error in the information part will not raise the error flag. However, as all single errors in the information part will be corrected. Thus that kind of injection is not important to us. Double errors and single errors not in the information part will always be detected. Therefore, an attacker will increase multiple-errors to maximize his chances of injecting an undetected error. Under the assumption that the attack only injects error vectors that manifest as multiple and not single or double errors, which are detected with a probability $1 - Q_{mc}$, the lower bound for the probability that an injection is detected is $\underline{P_{DET}} = 1 - Q_{mc} = 1 - b(b+1)(2^m-2)^{-1}$. ($P_{DET}$ is the probability that the attack at any moment is detected.)

Then the probability for the detection of attack, for consecutive A clock cycles, will be $P_A = 1 - \sum_{i=0}^{T-1} \binom{A}{i} P_{DET}^i (1 - P_{DET})^{A-i}$.

TABLE II
Error Detection and Correction Capabilities for SEC-DED AMC code

Number of errors	Error in parity	Errors in v_1	Errors in v_2 and/or v_3	Errors in v_4 [I]
Single	Detected	Corrected	Detected [II]	Detected
Double	Detected. No miscorrection.			
Multiple even	Detected with a probability $1 - Q_{V_{AMC}}$. [III] No miscorrection.			
Multiple odd	Detected with a probability $1 - Q_{mc}$. [IV] Miscorrected with a probability Q_{mc}.			

[I] If errors are located only in the v_4, no errors in the other parts of codeword c, these errors will always be detected.
[II] Here if we assume there is only a single error, then when the error is not in v_1, it is in v_2 or v_3 and can be corrected.
[III] $Q_{V_{AMC}}$ is the maximum error masking probability. $Q_{V_{AMC}} = (b+1)2^{-m}$
[IV] Q_{mc} is the error miscorrection probability. $Q_{mc} \leq b(b+1)2^{-m}$

(We assume that the attack detection events for different clocks are independent. Since the number of uncorrectable errors are continually monitored, when an attack is missed at a clock, the attack can still be detected at any following clock with the same probability P_A, if the attacker continuously injects errors.)

Thus the upper bound for T can be found for the following inequality for the given P_{A0}, A.

$$\begin{aligned} P_{A0} &\leq 1 - \sum_{i=0}^{T-1} \binom{A}{i} \underline{P_{DET}}^{i} (1 - \underline{P_{DET}})^{A-i} \\ &= 1 - \sum_{i=0}^{T-1} \binom{A}{i} (1 - Q_{mc})^{i} (1 - (1 - Q_{mc}))^{A-i}. \end{aligned}$$

For the proposed SEC-DED AMC code, $Q_{mc} = b(b+1)(2^m - 2)^{-1}$, then we have

$$P_{A0} \leq 1 - \sum_{i=0}^{T-1} \binom{A}{i} (1 - b(b+1)(2^m-2)^{-1})^{i} (b(b+1)(2^m-2))^{A-i}.$$

If we use the proposed AMC code as double-error-correcting code, then $Q_{mc} = 0.5b(b-1)m(b+1)(2^m - 2)^{-1}$, assuming that $(b^2(0.5(m+r_H-1))(b+1) < 0.5b(b-1)m(b+1))$. Then we have

$$\begin{aligned} P_{A0} \leq & 1 - \sum_{i=0}^{T-1} \binom{A}{i} (1 - 0.5b(b-1)m(b+1)(2^m-2)^{-1})^{i} \\ & (0.5b(b-1)m(b+1)(2^m-2)^{-1})^{A-i}. \end{aligned}$$

We also need to compute the *lower bound* for T, such that we will be able to differentiate between uncorrectable random errors and attacks. Otherwise, if there are T uncorrectable random errors in A consecutive clock cycles, we will disable the device, which is a false alarm. We would like to keep the false alarm probability below a given level P_{FA0}.

Assume that the bit distortion rate for a binary symmetrical channel representing random errors at the output of the memory is p, and the random numbers x are uniformly distributed. Denote the probability for the detection of uncorrectable random errors at any clock by P_R.

For a targeted false alarm probability P_{FA0} tolerated by the system, by selecting the minimal integer of T satisfying the following inequality, we can find the lower bound for the threshold T.

$$\begin{aligned} P_{FA0} &\geq 1 - \sum_{i=0}^{T-1} \binom{A}{i} \overline{P_R}^{i} (1 - \overline{P_R})^{A-i} \\ &\geq 1 - \sum_{i=0}^{T-1} \binom{A}{i} P_R^{i} (1 - P_R)^{A-i} \\ &= P_{FA}, \end{aligned}$$

where $\overline{P_R}$ is an upper bound for P_R.

For the upper bound for P_R for the proposed SEC-DED AMC code, we can use the following formula

$$\overline{P_R} = \sum_{i=2}^{n} \binom{n}{i} p^{i} (1-p)^{n-i} + (1-p)^{bm} \binom{2m+r_H}{1} p^{1} (1-p)^{2m+r_H-1},$$

where $n = k + m + m + r_H + 1 = (b+2)m + r_H + 1$ is the total number of bits in a codeword. (When calculating $\overline{P_R}$, we assume that all multiple errors are detected, and single error that is not in the information part is also detected.)

If we use the proposed AMC code as double-error-correcting code, we use the following formula To compute the upper bound for P_R,

$$\begin{aligned} \overline{P_R} = & \sum_{i=3}^{n} \binom{n}{i} p^{i} (1-p)^{n-i} \\ & + (1-p)^{bm} (2m + r_H) p (1-p)^{2m+r_H-1} \\ & + (1-p)^{bm} \binom{2m+r_H}{2} p^2 (1-p)^{2m+r_H-2}, \end{aligned}$$

where $n = k + m + m + r_H + 1 = (b+2)m + r_H + 1$ is the total number of bits in a codeword. (When calculating $\overline{P_R}$, we assume that all multiple (excluding double) errors are detected, and all single-bit and double-bit random errors that are not in the information part are also detected.)

If we would like to increase the probability P_A of detection of an attack, we should select T as small as possible. In this case, we may choose T equal to its lower bound.

Example 5.1: Consider a proposed SEC-DED AMC code, with distance 4, with parameters $m = 17$ and $b = 4$. The nonlinear encoding function is $f(y, x) = y_1 x \oplus y_2 x^2 \oplus y_3 x^3 \oplus y_4 x^4 \oplus x^7$. The total number of bits in a codeword is $n = 68 + 17 + 22 + 1 = 108$. In the worst case, at any clock $\underline{P_{DET}} = 1 - Q_{mc} = 1 - 30(2^{17} - 2)^{-1} \approx 0.99977$. If the bit distortion rate for the symmetrical binary channel is $p = 10^{-6}$, then the maximal probability for the detection of uncorrectable random errors at any clock is upper bounded by $\overline{P_R} \approx 4.0001 \times 10^{-5}$. Let us assume the attack lasts at least $A = 10$ consecutive clock cycles.

TABLE III
P_A AND P_{FA} VERSUS THRESHOLD T[I] FOR $A = 10$

Maximal $(-log_{10}(P_{FA}))$			Maximal $(-log_{10}(1 - P_A))$	T
$p = 10^{-4}$[II]	$p = 10^{-6}$ [II]	$p = 10^{-8}$[II]	$(Pr.$ of missing $)$	
1.40	3.40	5.40	36.40	1
3.15	7.14	11.14	31.76	2
10.64	11.11	11.14	27.47	3
5.12	15.00		23.40	4
9.59	15.35		19.52	5
12.06			15.80	6
14.68			12.23	7
15.95			8.84	8
15.95			5.63	9
			2.64	10

[I] For the extended (68,17,22) AMC code. Assume $A = 10$.
[II] $\overline{P_R} = 4.014 \times 10^{-3}$ for $p = 10^{-4}$;$\overline{P_R} = 4.000 \times 10^{-5}$ for $p = 10^{-6}$; $\overline{P_R} = 4.000 \times 10^{-7}$ for $p = 10^{-8}$.

Table III and Figure 3 present a relationship between the threshold T and the minimal probability for detection of an attack P_{A0} and the maximal probability for the false alarm P_{FA0}, based on parameters described above, i.e. $A = 10$, $\underline{P_{DET}} = 1 - Q_{mc} = 1 - 30(2^{17} - 2)^{-1} \approx 0.99977$, $\overline{P_R} \approx 4.0001 \times 10^{-5}$ and the code is an extended AMC code, with distance 4, with parameters $m = 17$ and $b = 4$.

For a given P_{A0}, we may find the corresponding probability for the missing and a value of T, which should be the upper bound of T. If we known the P_{FA0}, then we can find the lower bound of T from the table.

From Table III, we see that if we would like to achieve $P_{FA} \leq P_{FA0} = 10^{-10}$, T should be larger than or equal to 3 (with $p = 10^{-6}$). If we would like to achieve the probability for the missing for the attack $(1 - P_A) \leq (1 - P_{A0}) = 10^{-10}$, T should be smaller than or equal to 7. Consequently, we can select any T from the interval $T \in \{3, 4, 5, 6, 7\}$

By selecting $T = 3$, we are able to achieve the smallest probability for the missing for the attack, but the largest probability for a false

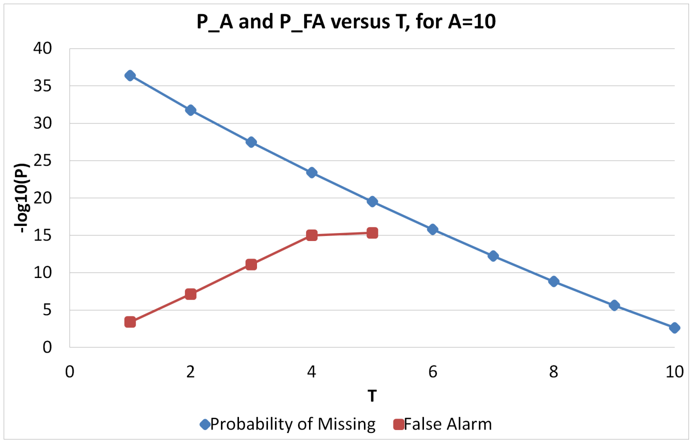

Fig. 2. P_A and P_{FA} versus T, for A=10 and $p = 10^{-6}$

alarm . The probability for the missing increases as T increases. Meanwhile, the false alarm probability decreases as T increases.

Example 5.2: We assume that the attack lasts at least $A = 6$ consecutive clock cycles. All other parameters are same as Example 5.1, i.e. $\underline{P_{DET}} = 1 - Q_{mc} = 1 - 30(2^{17} - 2)^{-1} \approx 0.99977$, $\overline{P_R} \approx 4.0001 \times 10^{-5}$ and the code is an extended AMC code, with distance 4, with parameters $m = 17$ and $b = 4$.

TABLE IV
P_A AND P_{FA} VERSUS THRESHOLD T^{I} FOR $A = 6$

Maximal $(-log_{10}(P_{FA}))$			Maximal $(-log_{10}(1-P_A))$	T
$p = 10^{-4\text{II}}$	$p = 10^{-6\ \text{II}}$	$p = 10^{-8\text{II}}$	$(Pr.$ of missing $)$	
1.62	3.62	5.62	21.84	1
3.62	7.62	11.62	17.42	2
5.89	11.89	11.62	13.39	3
8.41	15.48		9.62	4
11.20	15.48		6.10	5
14.37			2.86	6

I For the extended (68,17,22) AMC code. Assume $A = 6$.
II $\overline{P_R} = 4.014 \times 10^{-3}$ for $p = 10^{-4}$; $\overline{P_R} = 4.000 \times 10^{-5}$ for $p = 10^{-6}$; $\overline{P_R} = 4.000 \times 10^{-7}$ for $p = 10^{-8}$.

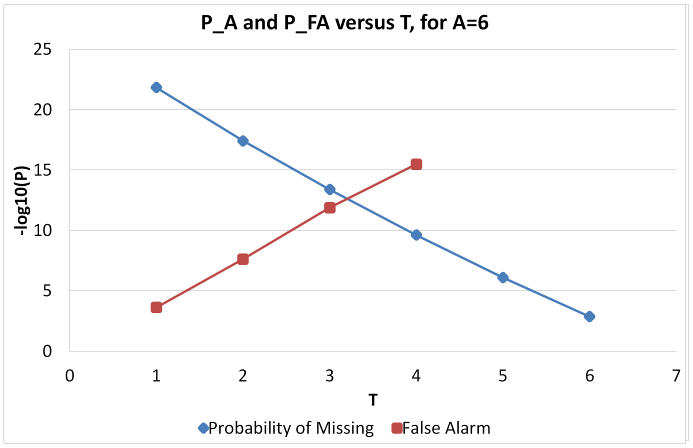

Fig. 3. P_A and P_{FA} versus T, for A=6 and $p = 10^{-6}$

Table IV and Figure 4 present a relationship between the threshold T and the minimal probability for detection of an attack P_{A0} and the maximal probability for the false alarm P_{FA0}, based on parameters described above.

From Table IV, we see that if we would like to achieve $P_{FA} \leq P_{FA0} = 10^{-10}$, T should be larger than or equal to 3 (with $p = 10^{-6}$). If we would like to achieve the probability for the missing for the attack $(1 - P_A) \leq (1 - P_{A0}) = 10^{-10}$, T should be smaller than or equal to 3. Consequently, the only choice for T is 3

We see that in Example 5.2, we select smaller A than in Example 5.1, but we achieve the same probability for the missing and probability for a false alarm while other parameters keep same.

VI. Conclusions

In this paper we show two constructions of nonlinear error correcting codes that can be used to build reliable and secure memories. The resulting memory architecture can tolerate both fault injection attacks and random errors. The presented codes can provide a guaranteed level of security as well as a satisfactory reliability. This is very important for memories used in cryptographic devices where both malicious attacks and errors introduced by mother nature cannot be ignored. The methodology used to distinguish between random errors and maliciously injected faults is also discussed.

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] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, 1998.

[5] R. Naseer and J. Draper, “Dec ecc design to improve memory reliability in sub-100nm technologies,” in *Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on*, 31 2008-sept. 3 2008, pp. 586 –589.

[6] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe, “Multi-bit error tolerant caches using two-dimensional error coding,” in *Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*, dec. 2007, pp. 197 –209.

[7] A. Dutta and N. Touba, “Multiple bit upset tolerant memory using a selective cycle avoidance based sec-ded-daec code,” in *VLSI Test Symposium, 2007. 25th IEEE*, may 2007, pp. 349 –354.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] ——, “Algebraic manipulation detection codes and their applications for design of secure communication or computation channels,” *Design, Codes and Cryptography*, 2012, submitted.

[14] 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.

[15] E. Trichina and R. Korkikyan, “Multi fault laser attacks on protected CRT-RSA,” *Workshop on Fault Diagnosis and Tolerance in Cryptography*, vol. 0, pp. 75–86, 2010.

[16] Z. Wang, M. Karpovsky, and A. Joshi, “Nonlinear multi-error correction codes for reliable mlc nand flash memories,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 7, pp. 1221 –1234, july 2012.

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