Reliable and Secure Memories Based on Algebraic Manipulation Detection Codes and Robust Error Correction

Shizun Ge
ECE, Boston University
Boston, MA, USA
Email: shizunge@gmail.com

Zhen Wang
Mediatek Wireless, Inc
Boston, MA, USA
Email: wang.zhen.mtk@gmail.com

Mark Karpovsky
ECE, Boston University
Boston, MA, USA
Email: markkar@bu.edu

Pei Luo
ECE, Boston University
Boston, MA, USA
Email: luopei@bu.edu

***Abstract*—The reliability and security of memories are crucial considerations in the modern digital system design. Traditional codes 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. Thereby, they may not be sufficient to protect memories against malicious attackers with strong fault injection capabilities and cannot correct unexpected errors with high multiplicities. The contribution of this paper is that we construct a new reliable and secure memory architecture based on robust Algebraic Manipulation Correction (AMC) codes. These codes can be used for correction of random errors and for detection of fault injection attacks. These codes can provide a guaranteed error detection probability for all errors even if both the user defined messages (data stored in the memory) and the error patterns are controllable by an attacker. The presented code can correct all single-bit errors in the information bits of the code. Moreover, these codes can be used to correct double-bit errors with high probabilities. The construction and the error correction procedures for the code will be described. The probability that an error can be successfully detected and/or corrected and the hardware overheads for the proposed memory architecture will be estimated. The presented approach is efficient for protecting security/reliability critical memories used to store the important information on the chip (e.g., a secret key in a cryptographic device).**

Keywords-Algebraic Manipulation Detection Code, Error Correction, Fault Injection Attack, Hardware Security

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.

The reliability of memory is a crucial consideration for today's digital devices. For some designs as much as 70% of the chip area is taken by the embedded memory [1], [2]. This large area of the chip is especially vulnerable to single-event-upsets (SEUs) caused by single, energetic particles like high-energy neutrons and alpha particles. SEU temporarily alters the state of the devices and results in soft errors. These errors are nondestructive and appear as unwanted bit flips in memory cells and registers. Continuing scaling of device features and performance increases the likelihood of errors, which makes the error models more unpredictable. As the speed of the devices becomes higher the relative size of the clock transition timing window increases and this makes devices more sensitive to SEU [3]. Similarly, decreased voltage levels for modern technologies make bit inversions more likely to occur [4].

The dangers of possible errors in memories resulting from SEUs are often mitigated with the use of linear single-error-correcting, double-error-detecting codes (SEC-DED). These codes have minimum Hamming distance four and are able to correct all single bit errors and detect all double bit errors. In the presence of multi-bit errors, however, the reliability of systems utilizing error protection architectures based on these codes may be questionable. For any linear SEC-DED codes with k information bits, the number of undetectable multi-bit errors is 2k. In addition to this, a huge number of multi-bit errors will be miscorrected. In the case where SEU results in multi-bit distortions with high probability, these codes may not be sufficient to provide a high reliability. Anomalies of systems caused by multi-bit upsets (MBU) have already been reported [5], [6].

The increase of the MBU rate in deep submicron technologies deteriorates the situation even further. In 65nm triple-well SRAMs with a thin cell architecture, the rate of multi-bit errors caused by neutron induced SEU increases by a factor of ten compared to that in 90 nm technologies nearly 55% of the errors due to neutron radiation were multi-bit errors [7]. Although there are mechanisms like bit interleaving [8] that can be used to minimize the error rate contribution of multi-bit errors, whether it is enough under such high MBU rate is still unknown. Moreover, the advantage of bit interleaving comes at a price of more layout constraints, which may result in larger power consumptions and longer access times. Thereby, memory protection architectures which can provide better protection against multi-bit errors than that based on classical linear codes are in demand.

Memories used in secure cryptographic devices not only suffer from random errors but are also vulnerable to errors injected by malicious fault injection attacks. It has been shown that the attacker can derive the secret key of the cryptographic devices thus break the security of the whole systems by injecting faults during encryption or decryption operations to force the devices working abnormally [9], [10].

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 [11], [12], [13]. The reliability and security of systems protected by linear codes largely depend on the accuracy of the expected error model. For memories used in cryptographic devices, the error model and the number of distorted bits introduced by the faults injected by the attacker are generally unpredictable due to the adaptive nature and the advanced fault injection methods available to an attacker. Thereby, the security of memories protected by linear codes cannot be guaranteed assuming the strongest attacker model.

As an alternative to linear codes, **robust codes** and their variants based on **nonlinear encoding functions** were proposed in [14], [15]. Different from linear codes, robust codes can provide nearly equal protection against all error patterns and are more suitable for applications where multi-bit errors are more probable or the error model is hard to predict. 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.

Intuitively, the limitation of robust code described above can be efficiently eliminated by introducing randomness into the encoding procedure of the code. Due to the fact that the random data are independent of the user information y, they can always be uniformly distributed. As a result, the assumption for robust codes that y is uniformly distributed is no longer required. Moreover, since the user has zero knowledge and no control of the random bits generated for each encoding operation, no matter how the attacker selects y and e, the probability that e is masked will be upper bounded by a number determined by the size of the set of possible random numbers. A coding technique based on adding to k information bits m random bits and r redundant bits, which overcomes the limitation of robust codes, is called strongly secure **algebraic manipulation detection (AMD)** code [16].

However, the constructions presented in [16] 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. The contribution of this paper is as following. In this paper, we propose new constructions of Algebraic Manipulation Correction (AMC) codes and efficient algorithms foe decoding for these codes. Comparing with our previous works [17], [18], [16], this code have a Hamming distance 4 and can correct all single-bit errors. Moreover, we describe a new error correction algorithm for the code which can also correct all double-bit errors with high probabilities. The proposed codes can provide a guaranteed level of security as well as a high level of reliability to the protected device, although the proposed scheme will require more redundancy, area and power. The problem of separating between single bit random errors and attacks for AMC codes is discussed in [17].

The rest part of the paper is organized as follows. The definitions of AMD codes and AMC codes are shown in Section II. In Section III-A, we describe the construction of the proposed AMC code and present the error correction algorithm for correcting random single and double errors. The hardware design and the analysis and comparison of overheads for the proposed codes are shown in Section IV.

II. Definitions

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**) [16] 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 $e^* = (e_y^*, e_x^*, e_f^*) \in K_S, e^* \neq \mathbf{0}$, where $\mathbf{0}$ is all zero vector, 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 $n = k + m + r$.

Definition 2.2: [19] 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

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

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 $Q_V = \max_y \max_{e \neq 0} Q_V(y,e)$.

The general architecture of a computational channel (device) protected by a (k,m,r) AMD code is shown in Figure 1, where RNG is a random number generator (either in software or hardware) and EDN is the error detection network.

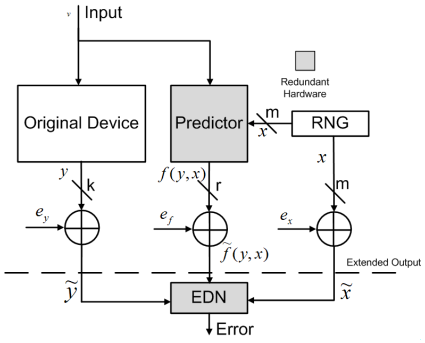

Fig. 1. Computation channel protected by a systematic (k,m,r) AMD code (original device and the predictor may be under attack).

The definition of AMD code can be found in [20], [16]. The existing AMD codes [20], [16] are designed for error detection and have Hamming distances less than 3. In this paper, we will construct Algebraic Manipulation Correction (AMC) codes that can be used not only for error detection but also for error correction. These codes can be used for design of reliable and secure memories where error correction is indispensable for restoring data distorted by natural effects such as soft errors. The formal definition of AMC codes is as follows.

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

III. Proposed Single-Error-Correcting AMC Codes

A. Construction of the proposed single-error-correcting AMC codes

In this Section, we will present the general construction of AMC codes and two error correction algorithms. The first algorithm can correct all single-bit errors and detect all double-bit errors. The second algorithm can correct not only single-bit errors but can also correct double-bit errors with high probabilities at the cost of higher hardware overhead.

Theorem 3.1: 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 $y = (y_1, y_2, \ldots, y_b)$; $y_i \in GF(2^m), (i = 1, 2, \ldots, b)$; $x \in GF(2^m)$; $f(y, x) \in GF(2^m)$ 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.

We note that the proposed AMC code is a combination of $(y, x, f(y, x))$ AMD code which provides for secure kernel $\{\mathbf{0}\}$ and (x, Px) Hamming code which provides for distance 3. This construction is similar to Vasil'ev nonlinear perfect code [21].

Remark 3.1: We may use any AMD encoding functions $f(y, x)$ described in [20], [16]. In general case, $y \in GF(2^k)$, where $k = sr$, and $x \in GF(2^m)$, where $m = tr$. In this case, $x = (x_1, x_2, \ldots, x_t)$ 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 3.1: 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 3.1 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 3.2: In a normal base Galois field [20], square operation can be achieved by the cyclic shift. As a result, $f(y, x)$ in Theorem 3.1 can be slightly modified to reduce its hardware complexity 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 $Q_{V_{AMC}} = 2^b(2^m - 2)^{-1}$.

B. 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 3.1, 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 3.1, 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 we only need to correct the errors in the information part $v_1 = y$. 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

$$\begin{aligned} S_{AMD} &= f(\tilde{y}, u) \oplus \tilde{v_4} \\ &= f(y \oplus e_1, x \oplus \pi e_1 \oplus e_2 \oplus \varepsilon) \oplus f(y, x) \oplus e_4. \end{aligned} \quad (3)$$

   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

Example 3.2: (Single Error Correction)
Consider a proposed AMC code with $b = 2$, $m = 3$. V_H is a $(6, 3, 3)$

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 $S_H \neq h_i$[I] $(\forall i)$	$\forall S_{AMD}$	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)

[I] 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.

Hamming code with $P = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$.

The encoding function is $f(y,x) = y_1 x \oplus y_2 x^2 \oplus x^5$. 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 3.1. 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

$$Q_{mc}(y,e) = |\{x | v_1 \neq \hat{v_1}, e \neq 0\}| 2^{-m}, \tag{4}$$

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

Theorem 3.2: Miscorrection Probability.

For the AMC code constructed by Theorem 3.1, the algorithm in Section III-B1 has a miscorrection probability $Q_{mc}(y,e)$, at most $b(b+1)(2^m-2)^{-1}$, $\max_{y;e\neq 0} Q_{mc}(y,e) \leq b(b+1)(2^m-2)^{-1}$.

The proof of this theorem is based on the analysis of the algorithm presented in Section III-B for the case $S_H \neq 0$, $S_H \neq h_i$ for each i (see Table I).

The AMC code for Theorem 3.1 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.

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. $Q_{mc} \leq b(b+1)2^{-m}$

Remark 3.3: 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 3.1.

For the straightforward concatenation approach $(y, x, f(y,x), P)$, the redundant parts are x, $f(y,x)$ and P, in which P is the Hamming redundant part for $(y, x, f(y,x))$ as the information part. Which means the redundant bits are $2m + \lceil log_2(mb + 2m + 1) \rceil$ for the straightforward concatenation approach. For the proposed architecture, the redundant parts are $\pi \oplus x$, xP and $f(y,x)$, the redundant bits number is $2m + \lceil log_2(m+1) \rceil$. For large b, the proposed architecture will save much more area than the straightforward concatenation approach.

C. Double Error Correction Algorithm and Estimations on the Probability of Miscorrection

1) Error correction algorithm: We note that the proposed AMC codes with overall linear parity bit can be used to correct double errors with a multiple-iteration algorithm. In this case the syndrome S_H for a Hamming code with distance 3 could be the sum of two columns h_{i_1} and h_{i_2} of the check matrix H, where i_1 and i_2 indicate the location of two errors.

For the proposed code, besides the syndrome for the Hamming code, we may use the syndrome for the AMD code to verify the location of the double error. After computing the syndrome, we can try all pairs (i_1, i_2) such that $h_{i_1} + h_{i_2} = S_H$ and if for one of them the corresponding AMD syndrome $S_{AMD} = 0$, it indicates that the double error is at the position i_1 and i_2.

We need to try all pairs of h_{i_1} and h_{i_2} with the same sum $S_H = h_{i_1} + h_{i_2}$. To achieve smaller number of iterations, we would like to make the number of pairs as small as possible. For a non-prefect Hamming code, we would like to select the check matrix H, such

that

$$\min_{H} \max_{S_H} |\{(i_1, i_2)|S_H = h_{i_1} \oplus h_{i_2}\}|, \tag{5}$$

where $\max_{S_H} |\{(i_1, i_2)|S_H = h_{i_1} \oplus h_{i_2}\}|$ indicates the maximum possible number of iterations.

We are able to pre-compute a lookup table for a fixed H for a Hamming code to simplify the locating of double errors for a given S_H.

Example 3.3: For a (11,7,3) shortened Hamming code with check matrix

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

we have $\max_{S_H} |\{(i_1, i_2)|S_H = h_{i_1} \oplus h_{i_2}\}| = 3$.

For another (11,7,3) shortened Hamming code with check matrix

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

we have $\max_{S_H} |\{(i_1, i_2)|S_H = h_{i_1} \oplus h_{i_2}\}| = 2$

Clearly, the second check matrix results in a smaller number of iterations to find the corrected error vector.

There are five parts (with the overall linear parity bit) in every codeword of the AMC code constructed as in Theorem 3.1, namely y, $\pi y \oplus x$, xP, $f(y, x)$ and the overall parity. For a codeword $v = (v_1, v_2, v_3, v_4, v_5)$ of the extended AMC code V_{AMC} constructed in Theorem 3.1, there are

$v_1 = y = (y_1, y_2, y_3, \ldots, y_b)$; $y_i \in GF(2^m)$, $i = 1, 2, 3, \ldots, b$;
$v_2 = \pi y \oplus x$; $\pi y,\ x,\ c_2 \in GF(2^m)$;
$v_3 = xP$; $xP \in GF(2^{r_H})$;
$v_4 = f(y, x)$; $v_4 \in GF(2^m)$;
$v_5 = \Pi(v_1, v_2, v_3, v_4)$

(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. v_5 is the overall linear parity bit.

Denote the error vector by $e = (e_1, e_2, e_3, e_4, e_5)$ and received message by $\tilde{v} = (\tilde{v_1}, \tilde{v_2}, \tilde{v_3}, \tilde{v_4}, \tilde{v_5})$, where $\tilde{v_i} = v_i \oplus e_i, i = 1, 2, 3, 4, 5$ and $e_1, \tilde{v_1} \in GF(2^{bm})$; $e_2, \tilde{v_2} \in GF(2^m)$; $e_3, \tilde{v_3} \in GF(2^{r_H})$; $e_4, \tilde{v_4} \in GF(2^m)$; $e_5, \tilde{v_5} \in GF(2)$. We assume that we only need to correct the errors in the information part $c_1 = y$. The decoding procedure can be divided into the following steps.

1) if the the overall parity indicates that there are errors with even multiplicities, go to step 2, otherwise it may be a single error.
2) calculate $(\tilde{u}, \tilde{v_3})$, where $\tilde{u} = \pi\tilde{v_1} \oplus \tilde{v_2}$
3) calculate $S_H = H(\tilde{u}, \tilde{v_3})^T$, the syndrome for the Hamming code.
4) use S_H as the input to a lookup table, then we can obtain some pairs of error locators ε_{i_1}, and $\varepsilon_{i_2}$where $\varepsilon_{i_1}, \varepsilon_{i_2} \in GF(2^m)$, for some i_1 and i_2, such that $S_H = h_{i_1} \oplus h_{i_2}$, where h_{i_1} and h_{i_2} are the i_1^{th} and i_2^{th} column respectively of the Hamming check matrix H.
   ε_{i_1} (as well as ε_{i_2}) is an m-bit vector with 1 in the only i_1^{th} (or i_2^{th}) bit and 0 in all other bits.
   Let $u = \tilde{u} \oplus \varepsilon_{i_1} \oplus \varepsilon_{i_2} = \pi\tilde{v_1} \oplus \tilde{v_2} \oplus \varepsilon_{i_1} \oplus \varepsilon_{i_2}$, where $u \in GF(2^m)$.
   If there are no such pairs in the look-up table, then no further steps need to be performed. The error is not correctable by this algorithm. Otherwise, go to the step 5.
5) calculate S_{AMD} as follows

$$\begin{aligned} S_{AMD} &= f(\tilde{y}, u) \oplus \tilde{v_4} \\ &= f(y \oplus e_1, x \oplus \pi e_1 \oplus e_2 \oplus \varepsilon_{i_1} \oplus \varepsilon_{i_2}) \oplus f(y, x) \oplus e_4. \end{aligned} \tag{6}$$

   If both $S_H = \mathbf{0}$ and $S_{AMD} = \mathbf{0}$, then there are no errors. Therefore, as long as $S_{AMD} = \mathbf{0}$, the correction procedure is completed. Otherwise go to the next step.
6) Compare S_{AMD} with $\varepsilon_{i_1} u^{j_1} \oplus \varepsilon_{i_2} u^{j_2}$, for all $j_1, j_2 \in \{1, 2, 3, \ldots, b\}$. If

$$S_{AMD} = \varepsilon_{i_1} u^{i_1} \oplus \varepsilon_{i_2} u^{j_2} \tag{7}$$

   for some $j_1, j_2 \in \{1, 2, 3, \ldots, b\}$, then the $y_{j_1}, y_{j_2} \in GF(2^m)$ of information $\tilde{v}_1 \in GF(2^{bm})$ of the codeword are distorted and the error in those parts are $\varepsilon_{i_1}, \varepsilon_{i_2} \in GF(2^m)$ respectively. That means $\hat{y}_{j_1} = \tilde{y}_{j_1} \oplus \varepsilon_{i_1}$, and $\hat{y}_{j_2} = \tilde{y}_{j_2} \oplus \varepsilon_{i_2}$,where $\hat{y}_{j_1}, \hat{y}_{j_2} \in GF(2^m)$ are the corrected messages.
7) If no j_1, j_2 are found, go back to step 4, try another pair $\varepsilon_{i_1}, \varepsilon_{i_2}$ for the same S_H.
8) If no $\varepsilon_{i_1}, \varepsilon_{i_2}$ satisfy (7), then the errors are not correctable for this algorithm.

Example 3.4: Consider an extended proposed AMC code with $b = 2$, $m = 7$. V_H is a $(11, 7, 3)$ Hamming code with

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The encoding function is $f(y, x) = y_1 x \oplus y_2 x^2 \oplus x^5$. The codeword is in the format of $v = ((y_1, y_2), \pi y \oplus x, xP, f(y, x), \pi_v)$, where y_1, y_2, x, $f(y, x) \in GF(2^7)$, $xP \in GF(2^4)$ and π_v is the overall parity. We select $z^7 \oplus z^3 \oplus 1$ as the generating polynomial for $GF(2^7)$, with the rightmost bit being the least significant bit.

Suppose $y_1 = (0000110)$, $y_2 = (0000011)$, $x = (0000010)$. Then we have $\pi y \oplus x = (0000110) \oplus (0000011) \oplus (0000010) = (0000111)$, $xP = (0110)$, and $f(y, x) = (0000110)(0000010) \oplus (0000011)(0000010)^2 \oplus (0000010)^5 = (0100000)$.

Thus, the original codeword is $v = (v_1, v_2, v_3, v_4, v_5) = ((0000110, 0000011), 0000111, 0110, 0100000, 1)$. Suppose there is a double error $e = ((0000001, 0001000), 0000000, 0000, 0000000)$ in the received message. Therefore, the distorted message is $\tilde{v} = ((0000111, 0001011), 0000111, 0110, 0100000, 1)$. We have $(\tilde{u}, \tilde{v_3}) = (\pi\tilde{v_1} \oplus \tilde{v_2}, \tilde{v_3}) = (0001011, 0110)$. $S_H = H(\tilde{u}, \tilde{v_3})^T = [P^T|I](\tilde{u}, \tilde{v_3})^T = (0101)^T$. The overall pairty shows that there is an error with a number of erroneous bits being even.

Since the rightmost bit is the least significant bit, we have the following lookup table for error locations.

S_H	i_1	i_2
0101	1	4
	2	7

Then let $i_1 = 1$ and $i_2 = 4$, then $\varepsilon_{i_1} = (0000001)$ and $\varepsilon_{i_2} = (0001000)$. We have $u = \tilde{u} \oplus \varepsilon_{i_1} \oplus \varepsilon_{i_2} = (0000010)$, and $S_{AMD} = \tilde{v_4} \oplus f(\tilde{v_1}, u) = (0100010)$. Let $j_1 = 1$ and $j_2 = 2$, we have $\varepsilon_{i_1} u^{j_1} \oplus \varepsilon_{i_2} u^{j_2} = (0000001)(0000010) \oplus (0001000)(0000100) = (0100010)$.

Therefore we know that the error ε_{i_1} locates at y_1 and ε_{i_2} locates at y_2. Actually, since $u = (0000010)$, if $i_1 = 1$ and $i_2 = 4$, then $S_{AMD} = (0100010) = \varepsilon_{i_1} u^{j_1} + \varepsilon_{i_1} u^{j_1}$ only if $j_1 = 1$ and $j_2 = 2$. If $i_1 = 2$, $i_2 = 7$, then $\varepsilon_{i_1} u^{j_1} + \varepsilon_{i_1} u^{j_1} = \varepsilon_2 u^{j_1} + \varepsilon_7 u^{j_2} \neq$

$S_{AMD} = (0100010)$ for any j_1, j_2. The corrected message is $\hat{v} = ((0000110, 0000011), 0000111, 0110, 0100000, 1)$.

In this example, we need at least 7 clock cycles to find the double error for the best case. Correspondingly, we will need $2+17*(14+3) = 291$ cycles to locate the double error for the worst case.

2) Estimations on the probability of miscorrection: In this section we estimate the upper bound for the probability that any error is miscorrected into a double error in the information part.

Theorem 3.3: For the algorithm of double error correction presented in Section III-C, the miscorrection probability Q_{mc} for any given pair e and y will be $Q_{mc} \leq (2^m - 2)^{-1} max(b^2 n_p(b+1),\ 0.5b(b-1)m(b+1))$.

The estimated Q_{mc} gives the upper bound for the miscorrection probability. Note that, a double error in the information part may be also miscorrected into another double error in the information part with the same miscorrection probability.

Example 3.5: (Miscorrection Probability) For an extended proposed AMC code with $b = 2$, $m = 7$, the check matrix for the $(11, 7, 3)$ Hamming code is the same as in Example 3.4. The encoding function is $f(y, x) = y_1 x \oplus y_2 x^2 \oplus x^5$, we have $n_p = 2$. Since $b^2 n_p(b+2) = 32 > 0.5b(b-1)m(b+2) = 28$, the maximum miscorrection probability is $Q_{mc} = (b^2 n_p(b+2))(2^m - 2)^{-1} = \frac{32}{126}$ for a given error e and a given information part y.

We conducted simulation experiments to estimate the probabilities of miscorrecting one double error into another one for $m = 2, b = 2$ and H as defined above with $n_p = 3$. All possible combinations of y and e are tested to see the number of random numbers x such that the errors are miscorrected.

The real miscorrection probability is $Q_{mc} = \frac{10}{126} \approx 0.08$, which is smaller than the estimated miscorrection probability is $Q_{mc} = \frac{32}{126}$.

Example 3.6: For an extended proposed AMC code with $b = 4$, $m = 17$, the check matrix for the $(22, 17, 3)$ Hamming code $H = [22, 21, \cdots, 2, 1]$, and encoding function $f(y, x) = y_1 x \oplus y_2 x^2 \oplus x^5$, we have $n_p = 6$.

Since $b^2 n_p(b+2) = 576 \leq 0.5b(b-1)m(b+2) = 612$, the maximum miscorrection probability is $Q_{mc} = (0.5b(b-1)m(b+2))(2^m - 2)^{-1} = \frac{612}{2^{17}-2}$ for a given error e and a given information part y.

A simulation was conducted to test the real miscorrection probability for a double error in the information part is miscorrected into another double error in the information part. $10,000$ random combinations of y and e have been tested to see the number of random numbers x such that the errors are miscorrected.

The evaluated miscorrection probability is $Q_{mc} = \frac{131}{2^{17}-2} \approx 0.001$, which is again much smaller than the estimated miscorrection probability $Q_{mc} = \frac{612}{2^{17}-2}$. We also note this miscorrection probability for double errors is small and converges to 0 very fast as the number of random bits m grows, thus the proposed code may be used for double-error-correction.

We note that compared with the direct construction $(y, x, f(y, x), P)$, the proposed architecture can protect the $f(y, x)$ at the same time while requires less redundant bits.

IV. Hardware Design of Reliable and Secure Memories Based on the Proposed AMC Codes

A. Hardware implementation

Figure 2 presents the architecture for memories protected by the proposed code in Theorem 3.1. In cryptographic applications, the m random digits can be generated by a random number generator (RNG) which is already integrated in most of the modern cryptographic devices. During a WRITE operation, the encoder loads the information bits and the random bits, then generates the redundant bits which are saved in the redundant memory block. The redundant bits are split into two parts, the linear part $(\pi y \oplus x, xP)$ which contains $m + r_H$ bits and the nonlinear part $f(y, x)$ which contains m bits. During a READ operation, the Error Correcting Network block computes the syndrome of the retrieved data and executes the error correction algorithm. If uncorrectable errors occur, ERR will be asserted and no correction will be attempted.

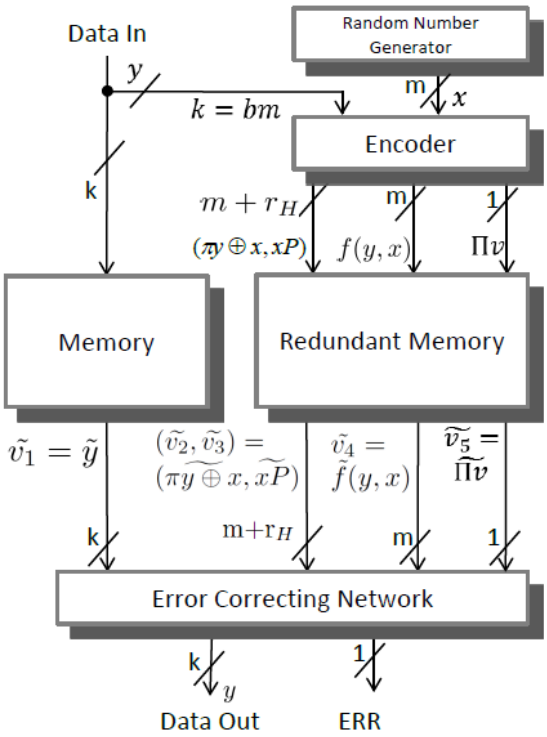

Fig. 2. Memory protected by the proposed code (the memory, redundant memory and the encoder may be under attack).

The encoder for the proposed code is presented in Figure 3. The Πv is the overall parity check of the system. The inputs to the encoder are the information bits y and the random number x. The encoder for the proposed AMC code contains the Hamming encoder and **the AMD encoder** to calculate $f(y, x)$. The AMD encoder requires b Galois field multipliers for $GF(2^m)$ to calculate the products of y and x^i and additional circuits to generate x^i.

The proposed code with distance 4 is a SEC-DED code, and only single error in the information part will be corrected. So if there are multiple errors or the single-bit error is not in the information part, the decoder sends the uncorrected messages to the output ports, and set the error flag (ERR). (ERR should be asserted iff there are uncorrectable errors.) The error flag signal will be used for the error handling, which is discussed in previous part of this paper. If there are even number errors, i.e., the overall parity bit c_5 is zero, no correction will be attempted.

To calculate the products between y_i and x^i in $f(y, x)$, for $i \in \{1, 2, \ldots, b\}$, b multipliers in $GF(2^m)$ are required. The computation of εu^i requires another b multipliers. We note that multipliers can be reused to reduce the area complexity at the cost of a larger decoding latency as more clock cycles are required. However, due to the reused hardware, a finite state machine should be introduced. If we do not reuse the multipliers, the architecture could be pipelined.

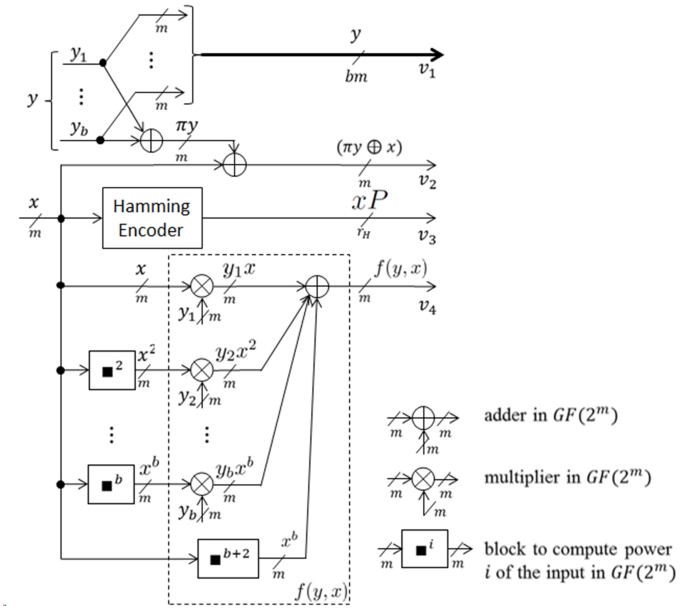

Fig. 3. Encoder without overall parity check bit for the proposed code

B. Comparison of the secure SEC-DED memories based on proposed codes with memories based on the known codes

In this section we compare the security levels in terms of the numbers of undetectable errors, sizes of security kernels and miscorrection probabilities, for the proposed codes V_{AMC} of distance 4 with additional overall check parity to the known nonlinear SEC-DED robust codes (extended Vasil'ev and extended Phelps) presented in [22]. The hardware overheads in terms of area, power and latency for encoders and decoders of different codes are also compared in this section. Transmission rates are also compared. All codes have distance 4. All the robust codes compared are $(39, 32, 4)$ codes [22].

For the proposed code, the number of information bits $k = bm = 35$, the number of random bits is $m = 7$, the degree of $f(y, x)$ is $b = 5$, the Hamming code used is a $(11, 7, 3)$ Hamming code. Overall parity is added to achieve Hamming distance four.

There are $35+7+7 = 49$ bits for the AMD part of the codes. The Hamming code used for error correction part is a $(55, 49, 3)$ code. Overall parity is also added to achieve Hamming distance four.

TABLE III
SIZES OF DETECTION KERNELS AND SECURITY KERNELS AND MISCORRECTION PROBABILITIES FOR DIFFERENT CODES

Codes	$\|K_D\|$ [I]	$\|K_V\|$ [II]	Q_{mc} [III]
Extended Hamming [IV]	2^{32}	$32(2^{32}-1)$	1
Extended Vasil'ev [IV]	2^{6}	$\approx 12(2^{32}-1)$	0.5
Extended Phelps [IV]	2^{27}	$32(2^{27}-1)$	1/16
Proposed code	0	0	30/126

[I] $|K_D|$ is the number of undetectable errors (size of detection kernel)
[II] $|K_V|$ is the size of security kernel
[III] Q_{mc} is the miscorrection probability
[IV] Data are obtained from [22]

Table III compares the sizes of detection kernels, i.e., the number of undetected errors, the sizes of security kernels and the miscorrection probability for the Hamming code, robust codes from [22], and the extended proposed code V_{AMC} with above parameters. It follows from the table that the proposed code is the best one providing good security for the strong attack model, as the code has zero size of the security kernel. The miscorrection probability for the proposed code is slightly worse than for the extended Phelps code, but it is still better than for the extended Vasil'ev code [22] and the Hamming code. Moreover, our code has no errors which are always undetected or miscorrected.

The encoder and the decoder for the proposed codes have been modelled in Verilog and synthesized in Cadence Encounter RTL Compiler with the Nangate 45nm Opencell library version v2009_07. The designs were placed and routed using Cadence Encounter. The latencies, the area overhead and the power consumptions of the encoders and the decoders were estimated using Concurrent Current Source (CCS) model under typical operation condition assuming a supply voltage of 1.1V and a temperature of 25 Celsius degree. The synthesis results for the encoder and decoder are shown in Table IV and Table V respectively. Those results were all obtained based on same simulation condition.

TABLE IV
SYNTHESIS RESULTS FOR ENCODERS

Architectures	Latency (ns)	Area (um^2)	Power (mW)
Extended Hamming [I]	0.290	282.2	0.2898
Extended Vasil'ev [I]	0.367	296.1	0.2916
Extended Phelps [I]	0.429	383.0	0.4728
Proposed code (smaller latency)	1.000	2246.6	3.444
Proposed code (lower power)	1.860	1573.9	1.044

[I] Data for the robust codes and the extended Hamming code are obtained from [22]

From Table IV, we can see that comparing to the extended Phelps code, the extended AMC code with distance four requires at least 161% increase in the latency. When optimizing the area and power, the latency increases to 433% of Extended Phelps Code encoder. The encoder of the extended proposed code also requires at least 310% more area and 121% more power than the encoder of the Extended Phelps code. This is the cost required to provide for the strong security.

We implemented the decoders for the proposed SEC-DED AMC codes with three profiles. One emphasizes the overall latency, another forces on the clock speed, and the third one requires the smallest area as well as power. To achieve the fastest clock speed, the decoder is pipelined. From Table V, we see that the smallest clock cycle of the decoder of our code is 1.096 ns, which is 64% more than the extended Phelps code, while 130% more area and 370% more power are used. On the other hand, sacrificing the clock speed, the decoder requires only 58% more area and 30% more power than the extended Phelps code. With 80% more area and 120% more power, we note that we can achieve the smallest overall latency, which is 1.971 ns.

From the results presented in this section one can see that the proposed AMC codes provide for better security (see Table III) but require larger overhead in latency, area and power (see Table III and Table II) than the known SEC codes (Hamming, Vasil'ev and Phelps).

C. Case studies for the proposed code

Figure 4 shows the transmission rate for different numbers of information bits for the extended Hamming code which is the same as the extended robust codes from [22] and the proposed SEC-DED

TABLE V
SYNTHESIS RESULTS FOR DECODERS

Architectures	Latency (ns)	Area (um^2)	Power (mW)
Extended Hamming [I]	0.538	620.3	0.7119
Extended Vasil'ev [I]	0.652	763.2	0.8340
Extended Phelps [I]	0.670	1799.8	1.774
Proposed code (smaller latency)	1.971	3272.9	4.056
Proposed code (faster clock)	1.096×2[II]	4215.6	8.49
Proposed code (lower power)	2.000×2[II]	2846.7	2.312
Double error correction for Proposed Code	0.917	11669.7	4.402

[I] Data for the robust codes and the extended Hamming code are obtained from [22]

[II] "×2" means there are two pipeline stages and the number in front indicates the clock speed.

AMC codes with distance 4 with three different parameter sets. One code uses $m = 7$ random bits, and (11,7,3) Hamming code. For another proposed code, we use $m = 17$ random bits and (22,17,3) Hamming code. The third proposed code use $m = 19$ random bits and (24,19,3) Hamming code. (We need that $2^m - 1$ will be a prime.) The degree b of nonlinear function of the proposed codes varies with the number of information bits.

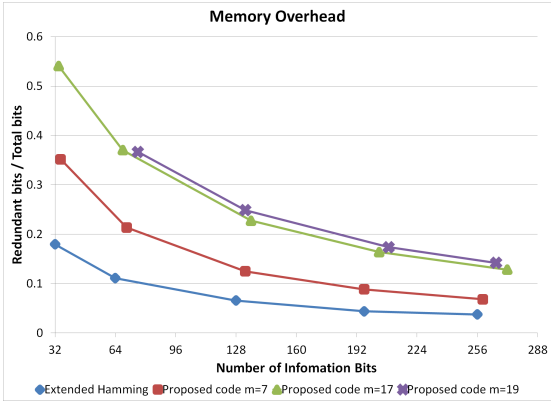

Fig. 4. Memory Overhead

The proposed codes require more redundant bits than the extended Hamming code, however as the number of information bits increases, the transmission rate of the proposed codes dramatically increases, since the number of redundant bits for the proposed codes depends on only m and does not depend on the number of information bits $k = bm$ as b increases. As k goes to infinity, the transmission rate of the proposed codes approaches one.

In order to decrease the probability for the missing, we can increase the number of random bits m. Moreover, in order to obtain a reasonable transmission rate, the degree b of the nonlinear function $f(y, x)$ should also be increased. Since $Q_{V_{AMC}} = (b+1)(2^m - 2)^{-1}$, the denominator increases exponentially with m, while the numerator increase linearly with b.

In the remainder of this section we consider the problem of the optimal selection of parameters m and b for a given number k of information bits for the proposed codes.

With the same m, as b increases, additional multipliers are required for $f(y, x)$ and the complexity for the encoder and the decoder are increasing. Additionally, an increase in the length k of the information part will result in an increase of the probability of multiple-bit random errors, which diminishes utility of SEC-DED codes. Therefore when choosing the parameters of the code, we should balance the transmission rate and the encoder and decoder's hardware complexities, security and reliability demands.

We propose the codes balancing these secure and reliable demands, with parameters listed in Table VI.

TABLE VI
PARAMETERS FOR THE PROPOSED CODES

k	m	$m + r_H$	b	$Q_{V_{AMC}}$	Q_{mc}
68	17	22	4	$6(2^{17} - 2)^{-1}$	$30(2^{17} - 2)^{-1}$
136	17	22	8	$10(2^{17} - 2)^{-1}$	$90(2^{17} - 2)^{-1}$
204	17	22	12	$14(2^{17} - 2)^{-1}$	$182(2^{17} - 2)^{-1}$
272	17	22	16	$18(2^{17} - 2)^{-1}$	$306(2^{17} - 2)^{-1}$
76	19	24	4	$6(2^{19} - 2)^{-1}$	$30(2^{19} - 2)^{-1}$
133	19	24	7	$8(2^{19} - 2)^{-1}$	$56(2^{19} - 2)^{-1}$
209	19	24	11	$12(2^{19} - 2)^{-1}$	$132(2^{19} - 2)^{-1}$
266	19	24	14	$16(2^{19} - 2)^{-1}$	$240(2^{19} - 2)^{-1}$

The hardware overhead for the encoders and the decoders for those SEC-DED codes are shown in Table VII and Table VIII respectively, under the same simulation condition as in Section IV-B. Note that the architectures here are not pipelined. (The pipelined version may use faster clock.) And we do not use the Horner scheme to implement the AMD encoding functions in order to achieve a smaller overall latency. We may also sacrifice the area and power to achieve a smaller latency.

TABLE VII
ENCODER OVERHEAD FOR THE PROPOSED CODES

Parameters of the codes	Latency (ns)	Area (um^2)	Power (mW)
m=17, b=4	2.623	5495	1.509
m=17, b=8	4.375	11375.2	3.968
m=17, b=12	4.403	16709.6	7.145
m=17, b=16	4.799	23578.8	11.23
m=19, b=4	3.181	7013.6	2.522
m=19, b=7	4.914	13788.1	6.052
m=19, b=11	4.945	20253.5	10.898
m=19, b=14	5.419	28587.9	17.128

Additionally, we note that the data output from the memory can be directly forwarded to other parts of the system, before the decoder generates its outputs. When errors are detected or corrected by the decoder, the processor can be stalled and the data can be re-fetched from the decoder. In this case, when no errors occur, the decoder does not affect the performance of the system in term of the latency. We may take advantage of this when our code is used for a memory, when the random access is frequently required.

V. CONCLUSIONS

In this paper, we show a reliable and secure memory architecture based on robust algebraic manipulation correction (AMC) codes. We describe the constructions of robust AMC codes, estimate their parameters (error detection and/or correction probabilities, etc) and

TABLE VIII
DECODER OVERHEAD FOR THE PROPOSED CODES

Parameters of the codes	Latency (ns)	Area (um^2)	Power (mW)
m=17, b=4	5.421	9040	1.761
m=17, b=8	7.455	17376	4.411
m=17, b=12	7.594	26389	7.487
m=17, b=16	8.127	35446	11.25
m=19, b=4	6.051	10500	2.372
m=19, b=7	6.666	19609	5.072
m=19, b=11	7.565	30568	9.68
m=19, b=14	8.886	38363	12.18

present the robust error correction algorithm for these codes. For the presented codes any error for any user defined message can be detected with a probability exponentially converging to one. Any repeating error (error with a high laziness) can be corrected with a probability converging to one as P_R or the number of redundant bits r grows. The area, power consumption and the latency for the encoder and the syndrome computation circuit are studied. The described architecture is suitable for the protection of the most security/reliability critical part (which is usually a small portion of the system) of the memory in many applications, where the error model is unpredictable (e.g., memories in cryptographic devices that may suffer from fault injection attacks) or the multi-bit error rate is high or difficult to estimate (e.g., nano-scale memories).

ACKNOWLEDGEMENT

The work of the third author is sponsored by the NSF grant CNS 1012910.

REFERENCES

[1] T.R.Halfhil, "Z-ram shrinks embedded memory," Microprocessor Report, Tech. Rep., Oct 2005.

[2] S.K.Moore, "Masters of memory," *IEEE Spectrum*, vol. 44, no. 1, pp. 45–49, Jan 2007.

[3] A. H. Johnston, "Scaling and technology issues for soft error rates," ser. 4th Annual Research Conference on Reliability, 2000.

[4] A. Eto, M. Hidaka, Y. Okuyama, K. Kimura, and M. Hosono, "Impact of neutron flux on soft errors in mos memories," in *Electron Devices Meeting*, 1998.

[5] S. Satoh, Y. Tosaka, and S. A. Wender, "Geometric effect of multiple-bit soft errors induced by cosmic ray neutrons on drams," June 2000.

[6] G. M. Swift, "In-flight observations of multiple-bit upset in drams," *IEEE Trans. Nuclear Science*, vol. 47, 2001.

[7] G. Georgakos, P. Huber, M. Ostermayr, E. Amirante, and F. Ruckerbauer, "Investigation of increased multi-bit failure rate due to neutron induced seu in advanced embedded srams," in *Symposium on VLSI Circuits Digest of Technical Paper*, 2007.

[8] J. Maiz, S. Hareland, K. Zhang, and P. Armstrong, "Characterization of multi-bit soft error events in advanced srams," in *IEEE Int'l Electronic Device Meeting*, December 2003, pp. 519–522.

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

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

[11] Y. Emre and C. Chakrabarti, "Memory error compensation techniques for jpeg2000," in *Signal Processing Systems (SIPS), 2010 IEEE Workshop on*, 2010, pp. 36–41.

[12] R. T. Chien, "Memory error control: beyond parity," *Spectrum, IEEE*, vol. 10, no. 7, pp. 18–23, 1973.

[13] Y. Bentoutou, "Efficient memory error coding for space computer applications," in *Information and Communication Technologies, 2006. ICTTA '06. 2nd*, vol. 2, 2006, pp. 2347–2352.

[14] 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.

[15] 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.

[16] 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.

[17] G. Shizun, W. Zhen, L. Pei, and K. Mark, "Secure memories resistant to both random errors and fault injection attacks using nonlinear error correction codes," in *Proc. Workshop on Hardware and Architectural Support for Security and Privacy, HASP 2013*, 2013.

[18] W. Zhen and M. Karpovsky, "Reliable and secure memories based on algebraic manipulation correction codes," in *On-Line Testing Symposium (IOLTS), 2012 IEEE 18th International*, 2012, pp. 146–149.

[19] 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*, ser. Lecture Notes in Computer Science, N. Smart, Ed. Springer Berlin / Heidelberg, 2008, vol. 4965, pp. 471–488.

[20] 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.

[21] J. L. Vasil'ev, "On nongroup close-packed codes," in *Probl.Kibernet.*, vol. 8, 1962, pp. 375–378.

[22] 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.