BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

NONLINEAR ROBUST CODES AND THEIR APPLICATIONS FOR DESIGN OF RELIABLE AND SECURE DEVICES

by

ZHEN WANG

B.S., M.S., Zhejiang University, 2006

Submitted in partial fulfillment of the

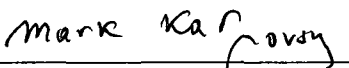
requirements for the degree of

Doctor of Philosophy

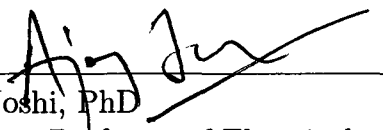
2011

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

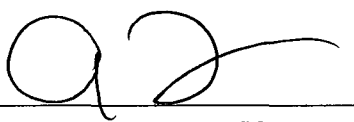
Approved by

First Reader 

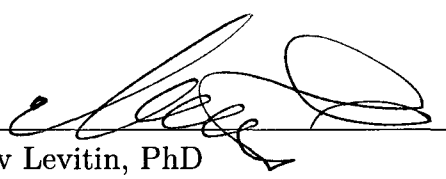
Mark G. Karpovsky, PhD
Professor of Electrical and Computer Engineering

Second Reader 

Ajay Joshi, PhD
Assistant Professor of Electrical and Computer Engineering

Third Reader 

Ari Trachtenberg, PhD
Associate Professor of Electrical and Computer Engineering

Fourth Reader 

Lev Levitin, PhD
Professor of Electrical and Computer Engineering

Acknowledgments

First of all, I would like to thank my advisor, Professor Mark Karpovsky for his instructions during the past four and half years. He turned me into a self-motivated, disciplined person capable of communicating with people and conducting research efficiently and independently using his patience, enthusiasm and unique vision and intuitive for research. Without his help, the completion of this dissertation would never become possible.

I would like to thank my co-advisor, Professor Ajay Joshi. Because of him, I started to pay more attention to practical industry applications of my theoretical works. I am also graceful to Professor Ajay Joshi for his helpful opinions of job hunting and his understanding and patience during my graduate study.

I would like to thank Professor Lev Levitin and Professor Ari Trachtenberg for spending time reading my dissertation and providing their invaluable suggestions and opinions. I also appreciate the help and guidance from Professor Berk Sunar in Worcester Polytechnic Institute. Several of my published papers are based on discussions with Professor Sunar.

I thank all stuff in the ECE department in Boston University. Thank you for providing me such a great environment with all necessary tools, computing resources, etc so that I can concentrate on the research.

Finally, I would like to thank my family for their continuous support of my life and my study. Thank my girl friend Yili Pan, who made me again an organized person after feeling lost and confused for several years. It is because of her that I finally become ready to face whatever challenge there is in my career and my life right now and in the future.

NONLINEAR ROBUST CODES AND THEIR APPLICATIONS FOR DESIGN OF RELIABLE AND SECURE DEVICES

(Order No.)

ZHEN WANG

Boston University, College of Engineering, 2011

Major Professor: Mark G. Karpovsky, PhD,
Professor of Electrical and Computer Engineering

ABSTRACT

Linear codes are widely used for error detection and correction in modern digital systems. These codes concentrate their error detecting and correcting capabilities on what are considered to be the most probable errors, which are typically errors of a small multiplicity. The reliability and the security of systems protected by these codes largely depend on the accuracy of the targeted error model. In many applications where the error model is hard to predict, the performance of linear codes cannot be guaranteed.

This work is on the development of special classes of codes – nonlinear robust codes with a given distance, multilinear codes and algebraic manipulation detection codes – for the design of secure cryptographic devices resilient to fault injection attacks and for the build of reliable memories. The primary difference between the proposed codes and linear codes is that the proposed codes provide nearly equal protection against all non-zero error patterns. As a result, the reliability and the security of the

protected devices can be guaranteed regardless of the accuracy of the error model. The advantages of the proposed codes over linear codes will become more significant if the same non-zero error pattern stays for several clock cycles.

The proposed codes are applied for various reliable and secure applications. The error detecting and correcting properties, the area, the power and the latency of the encoder and the decoder for designs based on the proposed codes are estimated and compared to those for designs based on linear codes. It is shown that adopting the proposed codes for the protection of modern digital devices can drastically reduce the number of errors undetected or miscorrected for all codewords thus increasing the reliability and the security of the system at the cost of a reasonable increase in the hardware overhead compared to protection mechanisms using linear codes.

Contents

List of Tables

List of Figures

List of Abbreviations

AES	.....	Advanced Encryption Standard
AMD	.....	Algebraic Manipulation Detection Codes
APN	.....	Almost Perfect Nonlinear
BCH	.....	Bose-Chaudhuri-Hocquenghem Codes
DFA	.....	Differential Fault Analysis
DRAM	.....	Dynamic Random Access Memory
EDN	.....	Error Detection Network
FSM	.....	Finite State Machine
GRM	.....	Generalized Reed-Muller Codes
MLC	.....	Multi-Level Cell
SEU	.....	Single Event Upset
MBU	.....	Multiple-Bit Upset
RS	.....	Reed-Solomon Codes
SEC-DED	.....	Sinlge-Error-Correcting, Double-Error-Detecting
SLC	.....	Single-Level Cell
SRAM	.....	Static Random Access Memory
TMR	.....	Triple Modular Redundancy

List of Notations

d	.....	Hamming distance
k	.....	Number of information bits
r	.....	Number of redundant bits
r_L	.....	Number of linear redundant bits
r_N	.....	Number of nonlinear redundant bits
M	.....	Number of codewords $M = \|C\|$
C	.....	Error control code
χ_c	.....	Characteristic function of C
c	.....	A codeword
e	.....	Error vector
$\tilde{c}$	.....	Distorted codeword
$GF(q^n)$	.....	Finite field
f	.....	Encoding function of systematic code
f_L	.....	Encoding function for linear redundant bits
f_N	.....	Encoding function for nonlinear redundant bits
P_f	.....	Nonlinearity of the function f
K_d	.....	Detection Kernel
K_c	.....	Correction Kernel
ω_d	.....	Dimension of the detection kernel.
$Q(e)$	.....	Error masking probability of e
$Q(y, e)$	.....	Error masking probability of e for given information bits y
$\|\|e\|\|$	.....	Hamming weight of e
$\|e\|_p$	.....	Mod p operation
$\oplus$	.....	Componentwise addition of q-ary vectors

Chapter 1

Introduction

Classical linear error detecting codes are designed for channels with specific error distributions. These codes concentrate their error detecting and correcting capabilities on what are considered to be the most probable errors, which are typically errors of a small multiplicity. However, in many environments and for many applications the assumptions that make the traditional methods efficient cannot be guaranteed, where the error distributions can be non-stationary or difficult to predict. In these situations, the reliability and the security of systems protected by linear error detecting codes cannot be guaranteed.

For example, the security of modern cryptosystems is threatened by side-channel attacks such as the timing analysis attacks (Kocher, 1996), the power analysis attacks (Kocher et al., 1999) and the fault injection attacks (Bar-El et al., 2006). Unlike other forms of side-channel attacks, fault based attacks are often active and hence adaptive. Due to the adaptive nature and the vast arsenal of fault injection methods and techniques available to the attacker, the model of errors injected by the attacker is impossible to predict. Any secure cryptographic devices based on linear error detecting codes can be easily compromised by injecting errors that are known to be undetectable by the used codes.

Continuous scaling of device features and performance causes an increased probability of errors and makes devices more susceptible to single-event-upsets (SEU), which complicates the analysis of the resulting error models. 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, for instance, the rate of multi-bit errors caused by neutron induced SEU increases by a factor of ten compared to that in 90nm technologies – nearly 55% of the errors due to neutron radiation were multi-bit errors (Georgakos et al., 2007). Moreover, due to the ubiquitous use of mobile computers, the environments in which devices operate and communicate change frequently and often drastically, which can cause non-stationary error distributions. Making an assumption about the error distributions in such cases can result in a poor reliability of the system due to the possibly incorrect error models.

Classical linear codes are also not suitable for lazy channels where faults have a high probability to manifest as the same nonzero error pattern for several clock cycles. Errors with a high laziness (probability that the error repeats) can occur in many hardware implementations. Faults in linear networks consisting of only XOR gates or fanout-free logic implementations will often result in internal faults manifesting themselves as repeating errors at the outputs of the devices. Failures in interconnect networks such as buses can also result in repeating errors. For such devices a single fault has a very high probability of manifesting itself in a constant error pattern regardless of the data it distorts.

Errors with high laziness can also occur in channels where an adversary is the cause of the malfunctions. Due to the limitation of the fault injection methodologies, the attacker cannot inject faults at each clock cycle (slow fault-injection mechanisms). Once faults are injected and an error is generated, the faults in most cases stay for several clock cycles before new faults can be injected and tend to manifest themselves as the same error patterns at the output of the device.

Repeating errors can be problematic for classical linear error detecting codes. When an error is not detected by one of the codeword, it will never be detected by

the code and the protected device will function erroneously without providing any detection of possible malfunctions.

To provide solutions to the limitation of classical linear error detecting codes, this work developed and analyzed three new families of error detecting codes – robust codes, multilinear codes and strongly secure algebraic manipulation detection (AMD) codes. These codes reduce or eliminate undetectable errors and provide nearly equal protection against all error patterns. As a result, the reliability and the security of the protected devices can be guaranteed regardless of the accuracy of the error models.

The first part of the dissertation describes the constructions of robust codes, multilinear codes and AMD codes and conducts theoretical analysis of the error detecting capabilities and the optimality of these codes. The definitions and bounds for robust codes are shown in Chapter 2. The constructions of robust codes based on nonlinear functions are presented in Chapter 3. Multilinear algebraic and arithmetic codes are described in Chapter 4.

The second part of the dissertation applies robust codes and multilinear codes for various reliable and secure applications. In Chapter 5, secure AES blocks, secure FSMs and secure multipliers resilient to fault injection attacks are built using the proposed codes. To protect cryptographic devices against the most advanced attackers, AMD are proposed in Chapter 6. In Chapter 7, minimum distance robust codes and partially robust codes are used for error detection and/or corrections in memories such as SRAMs, MLC NAND flashes, etc. All the designs are modeled in Verilog and synthesized in Cadence RTL design compiler. The overhead and the error detecting capabilities for designs based on different error detecting codes are analyzed and compared. The simulation results indicate that adopting robust and multilinear codes for applications such as secure cryptographic devices and reliable memory systems can results in much better reliability and security at the cost of a reasonable increase of

the hardware overhead compared to systems based on linear error detecting codes.

Summary of Contributions

In Chapter 2, different bounds for robust codes, systematic robust codes and minimum distance robust and partially robust codes are developed. A relationship between the worst case error masking probability and the nonlinearity of the encoding function is established. In Chapter 3, a number of optimal constructions of robust codes and their variations based on nonlinear functions are proposed. Existing constructions of perfect nonlinear Hamming codes such as Vasil'ev constructions and Phelps constructions are generalized to generate nonlinear error correcting codes with any length and any Hamming distance. The definitions and bounds for robust codes and partially robust codes were co-developed with Konrad Kulikowski. All the results related to minimum distance robust and partially robust codes were proposed by the author of the dissertation in (Wang et al., 2010a).

Variations of robust codes based on nonlinear functions are applied for the protection of AES linear blocks in Chapter 5. Related works have been published in (Karpovsky et al., 2007; Kulikowski et al., 2008b; Kulikowski et al., 2008c; Karpovsky et al., 2008a; Akdemir et al., 2011). The error correcting algorithms for minimum distance robust and partially robust codes are described in Chapter 7. These codes have been used for protecting SRAMs against single event upset (SEU) and multiple-bit upset (MBU) in (Wang et al., 2009c; Wang et al., 2010a) and for multi-bit error correcting for MLC NAND flash memories in (Wang et al., 2010b; Wang et al., 2011b).

Constructions of both algebraic and arithmetic multilinear codes are proposed in Chapter 4. Multilinear codes are adopted for the build of secure FSMs and secure multipliers resistant to fault injection attacks in Chapter 5. Related publications include (Wang et al., 2009b; Wang et al., 2009a; Wang and Karpovsky, 2010; Wang

et al., 2011c).

Bounds and constructions of algebraic manipulation detection codes for cryptography applications are developed in Chapter 6. It is shown that these codes are related to classical error detecting codes such as the Generalized Reed-Muller codes and the extended Reed-Solomon codes. These codes are used for the protection of Galois field multipliers for elliptic curve cryptography devices in (Wang and Karpovsky, 2011; Karpovsky and Wang, 2011).

The author of the dissertation also contributed in the design of synchronous and asynchronous power balanced gates (Kulikowski et al., 2008d; Kulikowski et al., 2008a); and in the analysis of the influence of nano-scale technologies (e.g. Carbon Nonotube Field-Effect Transistors (CNTFETs)) on the reliability of modern digital devices (Wang et al., 2011a).

Chapter 2

Definitions, Bounds and Optimality of Nonlinear Robust Codes

Classical error detecting codes concentrate their error detection and correction capabilities on a specific type of errors (e.g. errors with small multiplicities). These codes are characterized by three parameters – the length of the code (n), the number of codewords (M) and the Hamming distance (d). For these codes the design criterion is usually to maximize one of the three parameters while the other two are given. The optimality of classical error detecting codes is well studied in the community. A number of different bounds, e.g. Hamming bound, Plotking bound, Johnson bound, Linear programming bound, etc (MacWilliams and Sloane, 1998), have been proved and used to compare different codes in the past several decades. The best classical error detecting codes are those meeting at least one of the known bounds in the literature.

Nonlinear robust codes (Kulikowski et al., 2008b) are designed to provide nearly equal protection against all error patterns. Besides n, M and d, these codes are also characterized by the worst case error masking probability and the size of the detection (correction) kernel. A perfect robust code should have the smallest detection kernel and minimize the worst case error masking probability given all the other parameters.

In this Chapter, we start by defining the detection and the correction kernels of a code. Then the definitions of robust codes and their variations are presented. We

propose several bounds for robust codes, systematic robust codes and robust codes with minimum distance larger than 1. Several optimum constructions of robust codes and their variations meeting these bounds can be found in Chapter 3.

2.1 Kernels of Codes

Throughout the chapter we denote by "$\oplus$" the component-wise addition and "$\cdot$" the component-wise multiplication in a Galois field $GF(q^n)$. We denote by (n, k, d) a code of length n, dimension k and minimum distance d. Most of the results are presented for binary codes ($q = 2$). These results can be easily generalized for $q = p^s$, where p is a prime.

Definition 2.1.1 ***(Kernels of the code)*** *For any error correcting code $C \subseteq GF(2^n)$, the **detection kernel** K_d is the set of errors that are masked by all codewords.*

$$K_d = \{e | e \oplus c \in C, \forall c \in C\}. \tag{2.1}$$

It is easy to show that K_d is a linear subspace of C. (Let us denote the dimension of K_d by ω_d.) If C is linear, $K_d = C$.

*Denote by A the error correction algorithm for code C. Denote by E the set of errors that A attempts to correct. The **correction kernel** K_c is defined as follows:*

$$K_c = \{e | e \notin E, \forall c \in C, \exists e' \in E, A(e \oplus c) = A(e' \oplus c)\}. \tag{2.2}$$

Example 2.1.1 ***(Kernels of Linear Hamming Codes)*** *A $(n, n-\lceil log_2(n+1) \rceil, 3)$ linear Hamming code $C \subseteq GF(2^n)$ has minimum distance 3 and is able to correct all single bit errors. Denote by H the parity check matrix of C. An error e is undetectable if and only if e is a codeword ($He = 0$). Thereby the detection kernel K_d of a Hamming code is C itself. For single error correcting codes $E = \{e \big| ||e|| = 1\}$, where $||e||$ is the multiplicity of the error. A multi-bit error $e, ||e|| > 1$ will be miscorrected if and only if it has the same syndrome as some single bit error. So the correction kernel of Hamming code is $\{e | He = He_i^*\}$, where e_i^* is an error vector of Hamming weight one and He is the matrix multiplication in binary field. Obviously, K_d and K_c are disjoint. For perfect linear Hamming code, $K_d \bigcup K_c \bigcup E = GF(2^n)$.*

The main characteristic of traditional linear error detecting codes is that they concentrate their error detecting power on a small subset of errors which are assumed to be the most likely to occur. Typically, such codes concentrate their error detecting power on errors of a small multiplicity. They are designed to guarantee detection of all errors with a multiplicity less than d. Error detection beyond the minimum distance of the code is typically not a part of the design criteria and can be unpredictable and ineffective. While for some classes of errors the codes provide 100% protection, for a very large class of errors linear codes offer no protection for all messages. For any linear systematic error detecting code of length n and dimension k there are 2^k undetectable errors. Linear codes have the largest detection kernel K_d (the set of undetectable errors) of any class of systematic codes with the same n and k.

2.2 Robust Codes and Autocorrelation Functions

Autocorrelations of logic functions are powerful tools for the analysis and synthesis of digital hardware (Karpovsky et al., 2008b). These functions are analogous to the classical correlation functions employed extensively in telecommunications (Degtyaryov and Slyozkin, 2001; Sherman, 1956), theory of stochastic proceses (Lange, 1967) and are strongly connected to discrete transforms such as Walsh and Vilenkin-Chrestenson transforms (Karpovsky et al., 2008b). We first review some basic definitions of autocorrelation functions.

Definition 2.2.1 ***(Autocorrelation Function)*** *For a function $f : GF(q^k) \rightarrow GF(q^r)$, the autocorrelation $B_f(e)$ of f is defined as*

$$B_f(e) = \sum_{x=0}^{q^k-1} f(x)f(x \oplus e), \tag{2.3}$$

where $e \in GF(q^k)$, $\sum$ are integer additions and $\oplus$ is addition in $GF(q^k)$.

It is clear from (2.3) that $B_f(e)$ is a convolution-type transform of the original function f, with the addition of the variable x by e performed in $GF(q^k)$.

Autocorrelation functions can be expressed in terms of double Vilenkin-Chrestenson transforms (Karpovsky et al., 2008b).

Theorem 2.2.1 *(Karpovsky et al., 2008b) For $f : GF(q^k) \to GF(q^r)$, denote by S_f its Vilenkin-Chrestenson transform and S_f^{-1} the inverse Vilenkin-Chrestenson transform. Then*

$$B_f(e) = q^k S^{-1}_{(S_f S_f^*)}(e), \tag{2.4}$$

where S_f^* is the complex conjugate of S_f.

Theorem 2.2.1 is a direct analogue to the Wiener-Khinchin theorem in classical Fourier analysis, and for $q = 2$ is called dydaic Wiener-Khinchin theorem. It enables us to use fast algorithms for calculation of spectral transforms to compute autocorrelation functions and simplify our analysis of the error detection ability of codes using spectral methods. These fast algorithms can be found in (Karpovsky et al., 2008b).

Definition 2.2.2 ***(Characteristic Function)*** *The characteristic function of a code $C \subseteq GF(q^n)$, is a function $\chi_c : GF(q^n) \to \{0, 1\}$ defined as*

$$\chi_c(x) = \begin{cases} 1, & x \in C \\ 0, & x \notin C \end{cases} \tag{2.5}$$

Characteristic functions of codes are all Boolean functions. Their autocorrelations can be caculated as:

$$B_{\chi_c}(e) = \sum_{x=0}^{q^n-1} \chi_c(x)\chi_c(x \oplus e). \tag{2.6}$$

We will refer to $B_{\chi_c}(e)$ as the autocorrelation of a code C. From (2.6), we have $B_{\chi_c}(0) = |C|$. Nonzero error e is masked for message $x \in C$ if and only if $\chi_c(x) = \chi_c(x \oplus e) = 1$. Thus $B_{\chi_c}(e), e \neq 0$ is the number of codewords that will mask a given

error e. Assuming that codewords are equiprobable, the error masking probability $Q(e)$ for a fixed error e and a code C can be defined as

$$Q(e) = \frac{B_{\chi_c}(e)}{B_{\chi_c}(0)} = \frac{|\{x|x \in C, x \oplus e \in C\}|}{|C|}, \tag{2.7}$$

, which is the fraction of codewords that mask a given error e.

2.3 Definitions of Robust Codes and Their Variations

Definition 2.3.1 *The code C is **robust** iff $\max_{e \neq 0} Q(e) < 1$, or equivalently the detection kernel of the code contains only the zero vector $K_d = \{0\}$.*

For a robust code the error masking probability is bounded for nonzero errors. The worst case error masking probability of a robust code is determined by its autocorrelation function.

Definition 2.3.2 *A code $C \subseteq GF(2^n)$ is a R-robust code iff the autocorrelation of the characteristic function of the code, $B_{\chi_c}(e)$ is bounded by R for any $e \neq 0$.*

$$R = \max_{0 \neq e \in GF(2^n)} |\{x|x \in C, x \oplus e \in C\}| = \max_{0 \neq e \in GF(2^n)} \sum_{x=0}^{2^n-1} \chi_c(x)\chi_c(x \oplus e). \tag{2.8}$$

A graphic depiction of the definition of a robust code is shown in Figure 2·1. Let $C \subseteq GF(2^n)$, and $\tilde{C}_e$ be the set of all codewords of C shifted by an element $e \in GF(2^n)$. The code C is **R-robust** if for any $0 \neq e \in GF(2^n)$, the size of the intersection of the two sets C and $\tilde{C}_e$ is upperbounded by R.

The above defined robust codes have beneficial properties when worst case error masking probability of the codes is considered. By definition of an R-robust code there are at most R codewords which can mask any fixed error e. The worst case error masking probability for a R-robust code with M codewords is at most R/M, assuming all codewords are equi-probable. Thereby, robust codes have a predictable behaviour

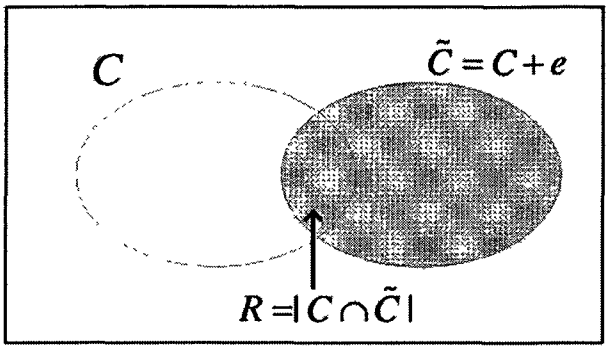


Figure 2·1: Definition of R-robust codes

in the presence of unpredictable error distributions as the worst case probability of masking of any error is bounded.

Most robust codes do not have a minimum distance larger than one and do not guarantee 100% detection probability for any subset of errors. A possible variant of the robust codes is to include a minimum distance into the design criteria.

Definition 2.3.3 *Let* $||e||$ *denote the multiplicity of an error* e*. A robust code where* $Q(e) = 0$ *for all* $||e|| < d, e \neq 0$ *is a* ***d-minimum distance robust code.***

Example 2.3.1 *Consider a 4-bit one hot code* $C = \{0001,\ 0010, 0100, 1000\}$*. It is easy to verify that for every nonzero error* $e \in GF(2^4)$*, there are at most two* $c \in C$ *satisfying* $c \oplus e \in C$*. Thereby,* $Q(e) = \frac{|\{c|c\in C, c\oplus e\in C\}|}{|C|} \leq 0.5$*,* $|K_d| = \{0\}$ *and* C *is robust. Moreover, for any single bit error* e*, there is no* $c \in C$ *satisfying* $c \oplus e \in C$*. The code* C *is a 2-minimum distance robust code.*

Minimum distance robust codes are robust codes with a minimum distance larger than one. Since these codes are robust they have no undetectable errors and the worst case error masking probability is bounded by $\max_{e\neq 0} Q(e) < 1$. However, unlike traditional robust codes they also provide a guaranteed 100% probability of detection of errors of small multiplicities ($||e|| < d$). These codes can be useful for providing the highest protection against the most likely or most dangerous threat while maintaining a detection guarantee in case of an unexpected behavior.

For some applications the error characteristics of robust codes can be considered too pessimistic. *Partially robust* codes and *minimum distance partially robust* codes (see Definition 2.3.4) allow for a tradeoff among robustness, decoding complexity and overheard, which fill the gap between the optimistic linear codes and pessimistic robust codes.

Definition 2.3.4 *A (n, k, d) code with a detection kernel smaller than 2^k is a* ***partially robust code****. If the code also has a minimum distance greater than one it is referred to as a* ***minimum distance partially robust code.***

Example 2.3.2 *The code $C = \{(x, p(x), f(x))\}$ where $x \in GF(2^{32})$, $f : GF(2^{32}) \rightarrow GF(2)$ is a perfect nonlinear function defined by $f(x = (x_1, x_2, ..., x_{32})) = x_1x_2 \oplus x_3x_4 \oplus ... \oplus x_{31}x_{32}$ and $p(x)$ is the linear parity function of x, is a $d = 2$ minimum distance robust error detecting code. For this code $Q(e) = 0$ when $||e|| = 1$ and $Q(e) \leq 0.5$ when $||e|| > 1$ (see Section 3.4.4).*

Partially robust codes reduce the number of undetectable errors while preserving some structures of linear codes which can be exploited to build efficient prediction hardware that generates redundant bits of a message. Like linear codes, partially robust codes still have undetectable errors (hence they are not completely robust). The number of undetectable errors is reduced by many orders of magnitude compared to that of the linear codes. For practical partially robust constructions, the number of undetectable errors can be reduced from 2^k to 2^{k-r} compared to a linear (n, k, d) code (Karpovsky and Taubin, 2004). The error masking probability of errors outside the detection kernel is upper bounded by

$$Q_{mc} = \max_{\{e|e \notin K_d\}} Q(e). \tag{2.9}$$

These errors will be ultimately detected assuming the error stays long enough and affects a number of different codewords.

2.4 Bounds, Optimality and Perfect Robust Codes

Based on the above definitions of the robust codes it is possible to derive the following main property for an R-robust code.

Property 2.4.1 *If the code C is R-robust then in the multiset $S_C = \{x_j \oplus x_i | x_i, x_j \in C, x_i \neq x_j\}$, any element appears at most R times.*

Robust codes are optimum if they have the maximum number of codewords M for a given R and and length n. From Property 2.4.1, a relation on R, n and M of the code can be established.

$$M^2 - M \leq R(2^n - 1). \tag{2.10}$$

Definition 2.4.1 ***(Perfect Robust Code)*** *A R-robust code with n bits and M codewords satisfying $M^2 - M = R(2^n - 1)$ is* ***perfect.***

It has been shown that perfect robust codes in binary field, which are the most important and practical for hardware design, exist only for even length and have the following parameters: $n = 2s, M = 2^{2s-1} \pm 2^{s-1}, R = 2^{2s-2} \pm 2^{s-1}$ (Beth et al., 1999). These codes can be constructed using the method presented in (Karpovsky and Nagvajara, 1989). Systematic codes, which are often more practical for many applications due to their seperation of data and check bits, cannot be perfect.

Theorem 2.4.1 *(Karpovsky et al., 2008a) For any systematic R-robust code with length n and k information bits, there are at least 2^{n-k} elements in $GF(2^n)$ which cannot be expressed as differences of two codewords.*

Proof For any systematic codeword $x = (x_1, x_2 = f(x_1))$ an error $e = (e_1, e_2)$ is masked iff $f(x_1 \oplus e_1) = x_2 \oplus e_2$. An error $e = (e_1 = 0, e_2 \neq 0)$ is never masked since $f(x_1) = x_2 \oplus e_2$ only iff $e_2 = 0$. An error that is never masked cannot be expressed as a difference of two codewords. Hence elements from $GF(2^n)$ of the form

$x = (0, x_2 \in GF(2^r))$, where $r = n - k$ cannot be expressed as a difference of two codewords. ■

Corollary 2.4.1 *There are no perfect systematic robust codes.*

When perfect robust codes are not available, the best possible codes which maximize M for a given n and R are referred to as *optimum* robust codes.

Definition 2.4.2 ***(Optimum Robust code)*** *Robust codes which have the maximum possible number of codewords M for a given length n and robustness R with respect to (2.10) are called* ***optimum****. For optimum codes adding any additional codewords would violate bound (2.10) and*

$$M^2 - M \leq R(2^n - 1) < M^2 + M. \tag{2.11}$$

Example 2.4.1 *Consider the following binary code $C = \{000, 001, 010, 100\}$ where $n = 3$. The mutiset S_C of all different pairs of codeword differences of the code is*

$$S_C = \{001, 001, 010, 010, 100, 100, \\ 011, 011, 101, 101, 110, 110\}.$$

Any nonzero element of $GF(2^3)$ appears at most two times in the multiset S_C, hence the code is 2-robust.

The code is not perfect since equality does not hold for (2.10). The code, however is an optimum $(3, 4, 2)$ Robust code. No other code can exist with the same n and R that has more codewords since 5 codewords would violate condition (2.10).

From Theorem 2.4.1, there are 2^{n-k} errors which will never be masked by any (n, k) systematic code. Thereby a more strict bound can be derived for systematic codes. In this case we have

$$M^2 - M \leq R(2^n - 2^{n-k}). \tag{2.12}$$

The best systematic codes with respect to M for given n and R should satisfy the equality in (2.12). These codes are also optimum regarding to 2.4.2.

2.5 Optimality of Systematic Minimum Distance Robust and Partially Robust Codes

The bounds presented in the last section do not depend on the distance of the code, thus are not precise for minimum distance robust codes. In this section, we present an exact upper bound for the size of the systematic minimum distance robust and partially robust codes in terms of Q_{mc}, n, k, d and the dimension of the detection kernel ω_d.

The redundant bits of a systematic code are generated by an encoding function $f : GF(2^k) \rightarrow GF(2^r), r = n - k$. In order to simplify the analysis, we distinguish between the encoding functions for linear and nonlinear redundant bits. Denote by r_L and r_N the number of linear and nonlinear redundant bits respectively. $r = r_L + r_N$. We represent a codeword c of a systematic code in the following format.

$$c = (x, f_L(x), f_N(x)), \tag{2.13}$$

where $f_L : GF(2^k) \rightarrow GF(2^{r_L})$ is the encoding function for linear redundant bits which can be implemented using only XOR gates, $f_N : GF(2^k) \rightarrow GF(2^{r_N})$ is the encoding function for nonlinear redundant bits which cannot be implemented using only XOR gates.

Theorem 2.5.1 *Denote by $r(d, n)$ the smallest possible number of redundant bits for a systematic code with minimum Hamming distance d and length n. For any (n, k, d) code C,*

$$2^k \leq Q_{mc}(2^n - 2^k 2^{r(d,n)} + (2^k - 2^{\omega_d})2^{r_N}) + 2^{\omega_d}, \tag{2.14}$$

where ω_d is the dimension of the detection kernel of C and $Q_{mc} = \max_{\{e|e \notin K_d\}} Q(e)$.

Proof Let $e = (e_1, e_2, e_3)$ be the error vector, where $e_1 \in GF(2^k), e_2 \in GF(2^{r_L}), e_3 \in GF(2^{r_N})$. We divide the errors into two classes as stated below.

1. $e_2 \neq f_L(e_1)$. These errors will be detected by the linear redundant bits of the code and are never masked. The number of errors in this class is $2^k(2^{r_L}-1)2^{r_N}$.

2. $e_2 = f_L(e_1)$. In this case an error $e = (e_1, e_2, e_3)$ is masked by a codeword $c = (x_1, x_2, x_3)$ iff there exists another codeword $c' = (x_1', x_2', x_3')$ such that

$$x_1 \oplus x_2' = e_1;$$
$$f_N(x_1) \oplus f_N(x_2') = e_3.$$

   Equivalently, $f_N(x_1 \oplus e_1) \oplus f_N(x_1) = e_3$. Errors in this class can be further divided into two classes.

   (a) If $e \in K_d$, it will be masked by all codewords of the code. The number of errors in this class is 2^{ω_d}.

   (b) If we can find an error $e' = (e_1', e_2', e_3')$ in K_d such that $e_1 = e_1', e_2 = e_2', e_3 \neq e_3'$, e will always be detected. The number of errors in this class is $2^{\omega_d}(2^{r_N} - 1)$.

   (c) All the other errors will be masked by no more than $2^k Q_{mc}$ codewords.

According to the above analysis, we have

$$\begin{aligned} 2^k &\leq Q_{mc}(2^n - 2^k(2^{r_L} - 1)2^{r_N} - 2^{\omega_d}2^{r_N}) + 2^{\omega_d} \\ &\leq Q_{mc}(2^n - 2^k 2^{r(d,n)} + (2^k - 2^{\omega_d})2^{r_N}) + 2^{\omega_d}. \end{aligned}$$

■

The function $r(d, n)$ can be estimated using existing bounds for error control codes that are extensively studied in the community. For example, the Hamming bound

and the Singleton bound (MacWilliams and Sloane, 1998). When $r(d,n)$ is derived from the Hamming bound, (2.14) is equivalent to

$$2^k \leq Q_{mc}(2^n - 2^k \sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i} + (2^k - 2^{\omega_d})2^{r_N}) + 2^{\omega_d}. \tag{2.15}$$

Definition 2.5.1 *A systematic minimum distance robust or partially robust code (n,k,d) satisfying the equality in (2.14) is* ***perfect.***

For the design of systematic minimum distance robust and partially robust codes, the best codes should have the maximum k given all the other parameters. When perfect codes are not available, the codes with maximum possible k when other parameters are fixed are called **optimum systematic minimum distance robust (partially robust) codes.**

Definition 2.5.2 *A (n,k,d) code which has the maximum possible k for given n, Q_{mc}, ω_d and d with respect to (2.14) is called* ***optimum.*** *For optimum codes increasing k will violate bound (2.14).*

Remark 2.5.1 *The optimality of minimum distance robust and partially robust codes that achieve the equality in bound (2.14) are twofold. First, it is perfect in a sense that it has the minimum number of redundant bits among all codes with distance d and length n. Second, it is perfect in a sense that it achieves the highest possible robustness with a given number of nonlinear redundant bits r_N. To be perfect in terms of bound (2.14), the following two conditions must be satisfied.*

1. *The total number of redundant bits $r = r_L + r_N = r(d,n)$;*
2. *$f_N(x)$ is a* ***perfect nonlinear function.***

The nonlinearity of a function $f_N : GF(2^k) \rightarrow GF(2^{r_N})$ can be measured using derivatives $D_a f_N(x) = f_N(x \oplus a) \oplus f_N(x)$. The nonlinearity can be defined by ((Carlet and Ding, 2004))

$$P_{f_N} = \max_{0 \neq a \in GF(2^k)} \max_{b \in GF(2^{r_N})} Pr(D_a f_N(x) = b), \tag{2.16}$$

where $Pr(E)$ denotes the probability of occurrence of event E. The smaller the value of P_{f_N}, the higher the corresponding nonlinearity of f_N. f_N is a perfect nonlinear function when $P_{f_N} = 2^{-r_N}$.

Example 2.5.1 *1. The nonlinear function $f_N : GF(2^{2st}) \rightarrow GF(2^t)$ defined by $f_N(x) = x_1 \bullet x_2 \oplus x_3 \bullet x_4 \oplus \cdots \oplus x_{2s-1} \bullet x_{2s}$, where $x_i \in GF(2^t), 1 \leq i \leq 2s$ and $\bullet$ is the multiplication in $GF(2^t)$ is a perfect nonlinear function with $P_{f_N} = 2^{-t}$ (Carlet and Ding, 2004).*

2. Let $f_N(x) : GF(2^n) \rightarrow GF(2^n)$ be a nonlinear function defined by $f_N(x) = x^3$, where x^3 is the cubing operation in $GF(2^n)$. For every $a, b \in GF(2^n), a \neq 0$, there are at most two x satisfying $D_a f_N(x) = b$. Thereby $P_{f_N} = 2^{-n+1}$. f_N is not a perfect nonlinear function because $P_{f_N} > 2^{-r_N}$. However, it has the smallest possible P_{f_N} among functions mapping $GF(2^n)$ to itself and is called almost perfect nonlinear (APN) (Carlet and Ding, 2004).

3. For and (n, k, d) linear systematic code C, let x be the information bits of the codeword and $H = [P|I]$ be the $r \times n$ parity check matrix in standard form, where I is the $r \times r$ identity matrix. The encoding function $f_L(x) = Px$ is linear and has $P_{f_L} = 1$.

More constructions of perfect and almost perfect nonlinear functions can be found in (Carlet and Ding, 2004).

2.6 Summary

Robust codes are designed to provide nearly equal protection against all error patterns. The worst case error masking probability $Q(e)$ of the code is determined by the maximum value of its autocorrelation function. The best robust codes should have no undetectable errors ($K_d = \{\mathbf{0}\}$) and minimize the worst case error masking probability ($Q(e)$ is minimized among all codes with the same length and the same size). Variations of robust codes, e.g. partially robust codes, minimum distance robust

and partially robust codes, are proposed for different applications and for tradeoffs between the hardware overhead and the reliability and security of the systems.

Several bounds for robust codes and their variations are presented. The definitions of perfect and optimum robust codes are described. Perfect robust codes can only be non-systematic and exist for very limited sets of parameters. Several constructions of optimum systematic robust codes and minimum distance partially robust codes will be shown in Chapter 3.

The definitions and bounds for robust codes and partially robust codes were co-developed with Konrad Kulikowski. All the results related to minimum distance robust and partially robust codes were proposed by the author of the dissertation in (Wang et al., 2010a).

Chapter 3

Systematic Robust Codes Based on Nonlinear Functions

In Chapter 2, we showed that perfect robust codes can only be nonsystematic codes. These codes correspond to well studied combinatorial structures known as difference sets and symmetric designs (Jungnickel and Pott, 1999). Despite the extensive research of the combinatorial structures it is still not known in the general case for what parameters such difference sets and hence perfect robust codes exist, and we note that the perfect robust codes based on the known difference sets have a high complexity of decoding (detecting for a given x whether $x \in C$ or $x \notin C$). A good summary of existing difference sets can be found in (Beth et al., 1999).

Systematic codes cannot be perfect. However, since the information bits and the redundant bits are separated, systematic codes usually have much lower encoding and decoding complexity thus are more practical for hardware design. There is a strong relationship between robust codes, nonlinearity, and nonlinear functions since all robust codes are nonlinear. The worst case error masking probability of systematic robust codes is determined by the nonlinearity of the encoding function of the codes.

In this Chapter, the constructions of several optimum systematic robust codes and minimum distance partially robust codes based on nonlinear encoding functions will be presented. These codes find applications in different areas such as the design of secure cryptographic devices resilient to advanced fault injection attacks (Chapter 5),

the build of multi-bit error tolerant reliable memory systems (Chapter 7), the error detection in lazy channels (Karpovsky et al., 2007), etc .

3.1 Optimum Systematic Robust Codes

We first review some basic definitions and properties of nonlinearity, a good survey of nonlinear functions can be found in (Carlet and Ding, 2004).

Let f be a function that maps elements from $GF(2^k)$ to $GF(2^r)$.

$$f : GF(2^k) \rightarrow GF(2^r) : a \rightarrow b = f(a). \tag{3.1}$$

The nonlinearity of the function can be measured by using derivatives $D_a f(x) = f(x \oplus a) \oplus f(x)$. Let

$$P_f = \max_{0 \neq a \in GF(q^k)} \max_{b \in GF(q^r)} Pr(D_a f(x) = b), \tag{3.2}$$

where $P_r(E)$ denotes the fraction of cases when E occurs. The smaller the value of P_f, the higher the corresponding nonlinearity of f. For linear functions $P_f = 1$.

Definition 3.1.1 *A function $f : GF(2^k) \rightarrow GF(2^r)$ has **perfect nonlinearity** if $P_f = 2^{-r}$.*

The parameters of systematic codes depend on nonlinearity of the encoding functions as shown by the next theorem.

Theorem 3.1.1 ***(Kulikowski et al., 2008b)*** *Let f be a function with nonlinearity P_f that maps $GF(2^k)$ to $GF(2^r)$ where $k \geq r$, the set of vectors resulting from the concatenation of $x_1, x_2 : (x_1, x_2 = f(x_1))$ where $x_1 \in GF(2^k)$ and $x_2 \in GF(2^r)$ forms a $(k + r, k)$ robust systematic codes with $R = P_f 2^k$.*

Proof The error $e = (e_1, e_2), (e_1 \in GF(2^k), e_2 \in GF(2^r))$ will be masked iff $f(x_1 \oplus e_1) \oplus f(x_1) = e_2, x_1 \in GF(2^k)$, which is exactly when $D_{e_1} f(x_1) = e_2$. ■

It is easy to verify that if the encoding function f is a perfect nonlinear function, systematic codes constructed as in Theorem 3.1.1 satisfy the equality in (2.12). These codes have flat total autocorrelation functions and have the maximum M for a given n and R. They are also optimum with respect to Definition 2.4.2.

Corollary 3.1.1 *A systematic robust code* $C = \{(x_1, x_2 = f(x_1))|x_1 \in GF(2^k), x_2 \in GF(2^r)\}$ *is optimum if the encoding function* f *is a perfect nonlinear function.*

Proof From Theorem 3.1.1 and Definition 3.1.1, if f is a perfect nonlinear function, the resulting code is a $(k+r, k)$ robust code with $R = 2^{k-r}$. The optimality of the code can be verified by using Definition 2.4.2. ■

Remark 3.1.1 *The nonlinearity of the encoding function* f *for systematic codes corresponds to the worst case error masking probability of the codes. We have:*

$$P_f = \max_{e=(e_1,e_2),e_1\neq 0} Q(e) = \max_{e\in GF(2^{k+r})} Q(e).$$

where $e_1 \in GF(2^k), e_2 \in GF(2^r)$.

The following two constructions are examples of optimum robust codes based on perfect nonlinear functions.

Construction 3.1.1 ***(Quadratic Systematic Code)*** *Let* $x = (x_1, x_2, \cdots, x_{2s}, x_{2s+1})$, $x_i \in GF(2^r), s \geq 1$. *A vector* $x \in GF(2^{(2s+1)r})$ *belongs to the code iff*

$$x_1 \cdot x_2 \oplus x_3 \cdot x_4 \oplus \cdots \oplus x_{2s-1} \cdot x_{2s} = x_{2s+1} \tag{3.3}$$

The resulting code is a $((2s+1)r, 2sr)$ *optimum robust code with* $R = 2^{(2s-1)r}$.

Proof The encoding function $f(x_1, x_2, \ldots, x_{2s}) = x_1 \cdot x_2 \oplus x_3 \cdot x_4 \oplus \cdots \oplus x_{2s-1} \cdot x_{2s}$ is a perfect nonlinear function with $P_f = 1/2^r$ (Carlet and Ding, 2004). From Theorem 3.1.1 the resulting code is $R = 2^{2sr}/2^r = 2^{(2s-1)r}$ robust code. ■

Example 3.1.1 ***(Robust Parity)****Methods based on linear parity check codes are often used for on-line error detection in combinational circuits (Lala, 2001). The linear 1-parity codes can detect all errors of odd multiplicities but offer no protection for errors of even multiplicities. For devices and environments where the error distributions are unknown or non stationary the linear 1-parity codes can result in unpredictable behavior in the presence of errors.*

As an alternative to the linear parity codes, the quadratic systematic robust codes defined in Construction 3.1.1 can be used. Taking $r = 1, q = 2$ *the encoding function becomes bent function (Carlet and Ding, 2004) and the resulting* $(2s + 1, 2s)$ *robust systematic code with* $R = 2^{2s-1}$ *(robust parity code) has the same redundancy as the linear parity codes. Unlike their linear counterparts, robust parity codes can detect any error with a probability of at least* $\frac{1}{2}$.

Applications of binary quadratic systematic codes for designing of memories with self-detection, for data transmission in noisy channel and for data verification can be found in (Karpovsky et al., 2007).

Perfect nonlinear functons, exist only for very limited sets of parameters. For the case of binary codes, there are no perfect nonlinear functions from $GF(2^k)$ to $GF(2^k)$ (Carlet and Ding, 2004; Yuan et al., 2006). Functions with optimum nonlinearity in this case are called almost perfect nonlinear (APN) functions (Maxwell, 2005), which have $P_f = 2^{-k+1}$. When f are APN functions in the binary field, the robust codes construted as in Theorem 3.1.1 have $R = 2$. These codes are not optimum.

Construction 3.1.2 ***(Robust Duplication Code)*** *Let* $x = (x_1, x_2), x_1, x_2 \in GF(2^r)$. *The robust duplication code* C *contains all vectors* $x \in GF(2^{2r})$ *which satisfy* $x_1^3 = x_2$ *where all the computations are in* $GF(2^r)$. *The code is a 2-robust code with* $n = 2r$ *and* $M = 2^r$.

Robust duplication codes can be a viable alternative to standard duplication techniques. Application of binary robust duplication codes to memories with self-error-detection and comparison between standard and robust duplication techniques can be found in (Karpovsky et al., 2007).

3.2 Modifications of optimum Systematic Robust Codes

New optimum codes can be constructed by modifying the old. We begin this section by introducing two modifications for systematic robust codes based on adding new codewords or deleting redundant bits.

Construction 3.2.1 ***(Augmented Code)*** *Let C be a (n,k) systematic R-robust code defined by the encoding function f. The augmented code $C^{(a)} = C \cup \{(\boldsymbol{0},\beta)|\beta \neq f(\boldsymbol{0}), \beta \in GF(2^{n-k})\}$ where $\boldsymbol{0}$ is the all-zeros vector in $GF(2^k)$, is a $(R+2)$-robust code with $M = 2^k + 2^{n-k} - 1$.*

Proof In the multiset $S_{\{(\mathbf{0},\beta\neq f(\mathbf{0}))\}}$ of vector differences of the set $\{(\mathbf{0},\beta \neq f(\mathbf{0}))\}$, each element is in the form $(\mathbf{0},\beta), \beta \neq f(\mathbf{0}), \beta \in GF(2^{n-k})$ and appears exactly $2^{n-k}-2$ times. From the Proof of Theorem 2.4.1 $(\mathbf{0},\beta) \notin S_C$. Thereby the two multisets are disjoint. In the multiset of differences of vectors of C and the additional vectors each element can appear at most two times. Hence, multiset of the augmented code $C^{(a)}$ contains each element in the form $(\mathbf{0},\beta)$ exactly 2^{n-k} times and all other elements at most $R+2$ times. ■

Remark 3.2.1 *Augmenting is only useful for codes with relatively large R. For augmented quadratic systematic codes, $n = (2s+1)r, M = 2^{2sr}+2^r-1$ and $R = 2^{(2s-1)r}+2$. When $s = 1$, these codes are optimum.*

Construction 3.2.2 ***(Punctured Code)*** *Let C be a (n,k) systematic R-robust code. Denote by C_p^* the punctured code obtained by deleting $p < (n-k)$ check bits from each codeword of C. C_p^* is a $(n-p,k)$ robust code with $R_p^* \leq 2^p R$.*

To prove the result we start with a Lemma.

Lemma 3.2.1 *Let $f : GF(2^k) \rightarrow GF(2^r)$ be a function with nonlinearity P_f. The punctured function $f^* : GF(2^k) \rightarrow GF(2^{r-1})$ formed by deleting one bit from the output of f has a nonlinearity of $P_{f^*} \leq 2P_f$. If f is a perfect nonlinear function then so is f^*.*

Proof The nonlinearity of the function f is defined as

$$P_f = \max_{0 \neq a \in GF(2^k)} \max_{b \in GF(2^r)} Pr(D_a f(x) = b) \tag{3.4}$$

, where $D_a f(x) = f(x \oplus a) \oplus f(x)$. Deleting one bit from the output of the function f will cause inputs which were previously mapped to outputs differing in the deleted bit to be remapped to the same output. Hence the derivatives of these elements which only differed in the deleted digit will be equal. The nonlinearity of f^* is therefore

$$P_{f^*} = \max_{0 \neq a \in GF(2^k)} \max_{i \in GF(2^{r-1})} 2Pr(D_a f(x) = (i, j)), \tag{3.5}$$

where (i, j) is the concanetantion of $i \in GF(2^{r-1})$ and $j \in GF(2)$. Since $Pr(D_a f(x) = (i, j)) \leq P_f$ for any a and (i, j) we have $P_{f^*} \leq 2P_f$.

When $f : GF(2^k) \rightarrow GF(2^r)$ is a perfect nonlinear function, $P_f = 2^{-r}$ and $P_{f^*} \leq 2^{-r+1}$. Since for a functions which maps $GF(2^k) \rightarrow GF(2^{r-1})$, the nonlinearity is lowerbounded by 2^{-r+1}, $P_{f^*} = 2^{-r+1}$ and f^* is perfect nonlinear. ■

The proof of Construction 3.2.2 easily follows from the above Lemma. Obviously, the resulting code will have the same length and the same number of codewords as the original code but worse robustness R as shown in Lemma 3.2.1.

Corollary 3.2.1 *Punctured robust codes formed by deleting $p < n - k$ check symbols from optimum systematic robust codes based on perfect nonlinear functions are also optimum.*

Proof The proof directly follows from Construction 3.2.2. ■

Punctured robust codes are still systematic codes. They are optimum as long as the encoding function of the original code is a perfect nonlinear function. Augmented robust codes, on the other hand, are not systematic codes. It is relatively difficult to

implement encoding and decoding for these codes. Augmented quadratic systematic code is an example of optimum nonsystematic robust codes.

We next show two modification methods based on the properties of autocorrelation functions that can generate new perfect robust codes from old ones.

Let $f(x) : GF(2^n) \rightarrow \{0,1\}$ be a boolean valued function and $\sum_{x=0}^{2^n-1} f(x) = M$. Let $\overline{f} = 1 + f \pmod 2$ be the inversion of f. Then

$$B_{\overline{f}}(e) = 2^n - 2M + B_f(e). \tag{3.6}$$

If f is the characteristic function $\chi_c(x)$ of a code $C \in GF(2^n)$, $\overline{f}$ will be the characteristic function $\overline{\chi}_c(x)$ of $\overline{C}$, which is the complement code of C containing all vectors in $GF(2^n)$ that do not belong to C. Thus we have:

Corollary 3.2.2 *Let C be a R-robust code of length n which has M codewords. Then its complement code $\overline{C}$ is a robust code of length n which has $2^n - M$ codewords. The robustness of $\overline{C}$ is $2^n - 2M + R$. Moreover, $\overline{C}$ is perfect if and only if C is perfect.*

Proof Recall that the robustness of the code C is

$$R = max_{e \in GF(2^n), e \neq 0} B_{\chi_c}(e), \tag{3.7}$$

where χ_c is the characteristic function of the code. Denote by $\overline{R}$ the robustness of $\overline{C}$. From theorem 3.2 we have:

$$\begin{aligned}
\overline{R} &= \max_{e \in GF(2^n), e \neq 0} B_{\overline{\chi}_c}(e) \\
&= \max_{e \in GF(2^n), e \neq 0} (2^n - 2M + B_{\chi_c}(e)) \\
&= 2^n - 2M + \max_{e \in GF(2^n), e \neq 0} B_{\chi_c}(e) \\
&= 2^n - 2M + R
\end{aligned}$$

Obviously, $\overline{C}$ will have "flat" autocorrelation if and only if C does. ■

We next give an example of modifications which do not generate codes with new parameters but only change the structure of the code. This method is due to the fact that autocorrelation functions are invariant to the affine transformation of variables.

Theorem 3.2.1 ***(Karpovsky et al., 2008b)*** *Let $f : GF(2^k) \to GF(2^r)$ and $\sigma = (\sigma_{ij})$ a binary matrix $(i, j = 0, 1, \cdots, k-1)$, $|\sigma| \neq 0$ ($|\sigma|$ denotes the determinant of σ). τ is a binary vector of length k. Denote by $\phi(x) = f(\sigma \otimes x \oplus \tau)$, where $\otimes$ is the matrix multiplication and "$\oplus$" is the componentwise addition over $GF(2)$. Then*

$$B_\phi(x) = B_f(\sigma \otimes x). \tag{3.8}$$

If $f = \chi_c(x)$ is the characteristic funciton of a code $C \in GF(2^n)$ with codewords $x = (x_1, x_2, \cdots, x_n)$, the by the above theorem the affine transformations of $(x_1, x_2, \cdots, x_n)$ do not change the "value distribution" of the autocorrelation and thus do not change R. Thereby we have

Corollary 3.2.3 *Let $x = (x_1, x_2, \cdots, x_n)$ be the codewords of a R-robust code of length n and size M, σ be a $n \times n$ nonsingular matrix and τ be a vector of length k over $GF(q)$. The new code constructed by applying an affine transformation $\sigma \otimes x \oplus \tau$ to x is still a R-robust code of the same length and the same size.*

Although not able to generate codes with new parameters, the above modification method enbles us to find codes that are easier to implement in hardware.

Example 3.2.1 *Consider an optimum systematic robust codes with $n = 5, M = 2^4$ and $R = 2^3$ defined by the following encoding function*

$$x_1x_2 \oplus x_2x_3 \oplus x_1x_3 \oplus x_3x_4 = x_5, x_i \in GF(2). \tag{3.9}$$

To implement this code in hardware, we need 4 2-input AND gates and 3 2-input XOR gates for the encoder. Let σ be

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

After applying the linear transformation $\sigma \otimes x$ to x, the encoding function becomes $x_1x_2 \oplus x_3x_4 = x_5, x_i \in GF(2)$, which is more organized and easier to implement. Compared with the original code, only two 2-input AND gates and one 2-input XOR gates are needed for the encoder, which cuts down the hardware redundancy by more than 50 %. The new code is a quadratic systematic code and has exactly the parameters as the old one.

3.3 Partially Robust Codes

Robust codes have a higher complexity of encoding and decoding than classical linear codes. The quadratic systematic codes with $n = (2s+1)r, k = 2sr$ and $R = 2^{(2s-1)r}$ in binary field from Construction 3.1.1 require s r-bit multipliers and $s-1$ r-bit componentwise additions. Assuming a r-bit multiplier requires r^2 two-input gates the encoder for these systematic quadratic code can be implemented with $sr^2 + r(s-1)$ 2-input gates.

As a tradeoff between robustness and the hardware overhead for computational devices, *partially robust codes* were introduced in (Karpovsky and Taubin, 2004). These codes combine linear and nonlinear mappings to decrease the hardware overhead associated with generation of check bits. The encoding of systematic partially robust code is performed first by using a linear function to compute the redundant check r check symbols followed by nonlinear transformation. The use of the linear code as the first step in the encoding process typically results in hardware savings in the encoder or predictor since the nonlinear function needs to only be computed based on the output of the linear block, which has length r that is much shorter than n. The application of the nonlinear transformation reduces the number of undetectable

errors thus increasing the robustness of the linear codes.

Construction 3.3.1 ***(Partially Robust Codes (Kulikowski et al., 2008b))*** *Let $f : GF(2^r) \rightarrow GF(2^r)$ be a nonlinear function with $P_f < 1$ and let $H : GF(2^k) \rightarrow GF(2^r)$, $r \leq k$ be a linear onto function. All vectors in the form $(x, f(H(x)))$ form a partially robust code in $GF(2^{k+r})$. The set of undetectable errors is a 2^{k-r} subspace of $GF(2^{k+r})$, $2^k - 2^{k-r}$ errors are detected with probability 1, and remaining $2^{k+r} - 2^k$ errors are detected with probability at least $1 - P_f$.*

Proof Error $e = (e_1, e_2), e_1 \in GF(2^k), e_2 \in GF(2^r)$ is masked if and only if

$$f(H(x \oplus e_1)) = f(H(x)) \oplus e_2, \tag{3.10}$$

or

$$f(H(x) \oplus H(e_1)) = f(H(x)) \oplus e_2. \tag{3.11}$$

If $H(e_1) = e_2 = 0$, (3.11) is always satisfied and the error will be masked with probability 1. Since H is a linear onto function, the number of errors $e = (e_1, e_2)$ (including 0) satisfying $H(e_1) = e_2 = 0$ is 2^{k-r} and these errors form a $k-r$ dimensional subspace of $GF(2^{k+r})$.

If $H(e_1) = 0$, $e_2 \neq 0$, then (3.11) will never be satisfied. These errors will be detected with probability 1. The number of these errors is $2^k - 2^{k-r}$.

If $H(e_1) \neq 0$, from (3.2), we know that given $e = (e_1, e_2)$ there are at most $P_f 2^r$ values of $H(x)$ satisfying (3.11). Because H is a linear onto function, for each value of $H(x)$, there are 2^{k-r} possible solutions for x. Thereby errors in this class will be masked no more than $P_f 2^r \cdot 2^{k-r} = P_f 2^k$ times. They will be detected with probability at least $1 - \frac{P_f 2^k}{2^k} = 1 - P_f$. ■

Example 3.3.1 *The code $C = \{(x, (Px)^3)\}$ where $x \in GF(2^{32})$, P is a 32 by 6 encoding matrix of a shortened $(38, 32)$ Hamming code, and the cubing operation is over $GF(2^6)$, is a binary partially robust code with $\omega_d = 26$ and $\max_{e \notin K_d} Q(e) = 2^{-5}$. C has the same number of redundant bits as the $(38, 32)$ shortened Hamming codes but much less undetectable errors. (For the $(38, 32)$ shortened Hamming code, $\omega_d = 32$.)*

The number of undetectable errors is reduced from 2^k to 2^{k-r} compared to the linear code with the same redundancy. The combination of a linear functions simplifies the prediction complexity for devices with linear or partially linear functions.

Application of partially robust codes for error detection in memories have been discussed in (Karpovsky et al., 2007). Partially robust codes with $k = 128$ and $r = 32$ have been used in (Karpovsky et al., 2004) for design of private key security devices based on Advanced Encryption Standard (AES) resistent to fault injection attacks. Implementation of this approach resulted in about 80% hardware overhead.

3.4 Minimum Distance Partially Robust Codes

For applications where the error model may be imprecise but errors with small multiplicities are more probable; or applications where not only error detecting but also error correcting is required, the distance of the code should be larger than 1. In this case minimum distance robust and partially robust codes can be a promising alternative to linear error correcting codes.

3.4.1 Vasil'ev Codes and Their Generalizations

Many known constructions of nonlinear codes are minimum distance partially robust codes. They have a minimum distance larger than one and have much fewer undetectable errors than linear codes. Such codes can even be perfect with respect to the classical Hamming bound (MacWilliams and Sloane, 1998). The first perfect nonlinear Hamming code was constructed by Vasil'ev in (Vasil'ev, 1962) and was generalized by Mollard in (Mollard, 1986). We first review the basic construction of the Vasil'ev code.

Theorem 3.4.1 ***(Vasil'ev Code(Vasil'ev, 1962))*** *For $u \in GF(2^m)$, let $p(u)$ be the linear parity check of u. Let V be a perfect not necessarily linear Hamming code of length $m = 2^r - 1$ with $k_V = m - r$ information bits. Let $f : V \to \{0, 1\}$ be an arbitrary*

nonlinear mapping such that $f(\mathbf{0}) = 0, \mathbf{0} \in GF(2^m)$ *and* $f(v) \oplus f(v') \neq f(v \oplus v')$ *for some* $v, v' \in V$. *The code* C *defined by*

$$C = \{(u, u \oplus v, p(u) \oplus f(v)) | u \in GF(2^m), v \in V\} \tag{3.12}$$

is a perfect nonlinear Hamming code.

Corollary 3.4.1 *Vasil'ev code is a* $(2m + 1, 2m - r, 3)$ *partially robust code with* $\omega_d = m$ *and* $Q_{mc} = P_f$ *where* P_f *is the nonlinearity of* f. *The code is perfect with respect to bound (2.14).*

Proof Let H be the check matrix of V. An error $e = (e_1, e_2, e_3)$ where $e_1, e_2 \in GF(2^m)$ and $e_3 \in GF(2)$ is masked if and only if $H(e_1 \oplus e_2) = \mathbf{0}$ and $f(v \oplus e_1 \oplus e_2) \oplus f(v) \oplus p(e_1) \oplus e_3 = 0$. The errors can be divided into four classes as follows.

1. $e_1 = e_2$ and $p(e_1) = e_3$, the error will always be masked. The number of errors in this class is 2^m;

2. $e_1 = e_2$ but $p(e_1) \neq e_3$, the error will always be detected. There are 2^m errors belonging to this class;

3. $H(e_1 \oplus e_2) = \mathbf{0}$ but $e_1 \neq e_2$, the error masking probability depends on the nonlinear function f; In the worst case, a specific error will be masked by $P_f \times |C|$ codewords. The number of errors in this class is $2^{m+1}(2^{m-r} - 1)$;

4. $H(e_1 \oplus e_2) \neq \mathbf{0}$. The error in this class will always be detected. The number of errors is $2^{m+1}(2^m - 2^{(m-r)})$.■

Vasil'ev codes are perfect single error correcting codes and have the same parameters as linear Hamming codes. The basic construction of Vasil'ev code can be further generalized as follows. The following Theorem can be proved in a similar way to the proof for Theorem 3.4.1 and Corollary 3.4.1.

Theorem 3.4.2 ***(Generalized Vasil'ev Code)*** *For $u \in GF(2^a)$, let $p(u)$ be the linear parity check of u. Let V be a $(m, k_V, 3)$ not necessarily linear Hamming code with $r = m - k_V$ redundant bits. without loss of generality, assume that the first k_V bits in any codeword of V are information bits. Denote by $v = (y, z)$, $y \in GF(2^{k_V}), z \in GF(2^r)$ the codewords of V. Select $f : GF(2^{k_V}) \rightarrow GF(2)$ to be an arbitrary mapping such that $f(\mathbf{0}) = 0$ and $f(y) \oplus f(y') \neq f(y \oplus y')$ for some $y, y' \in GF(2^{k_V})$. The code C defined by*

$$C = \{(u, (u, \mathbf{0}) \oplus v, p(u) \oplus f(y))\}, \tag{3.13}$$

where $u \in GF(2^a), \mathbf{0} \in GF(2^{m-a}), 0 < a \leq m, v \in V$ is a $(a + m + 1, a + k_V, 3)$ code with $\omega_d = a$ and $Q_{mc} = P_f$. C is optimum with respect to bound (2.14). Adding one more overall linear parity bit to C will result in a nonlinear SEC-DED code with the same ω_d and Q_{mc} as C and minimum distance four, which we call the ***extended Vasil'ev code.***

The significance of Theorem 3.4.2 is twofold. First, it can generate robust SEC-DED codes of arbitrary lengths. These codes have the same number of redundant bits as the best linear SEC-DED codes in terms of the number of redundant bits but much smaller number of undetectable multi-bit errors and are more suitable for applications where the MBU rate is high. Second, it allows a tradeoff between robustness and hardware overhead. Generally speaking, smaller a results in increased robustness of the code but requires more hardware for the encoder. By carefully selecting a and m, we can construct codes for situations that have different requirements for robustness and the hardware overhead.

Example 3.4.1 *1. Let $a = 16$ and V be a $(21, 16, 3)$ shortened Hamming code. Select f to be the same nonrepetitive quadratic function as in Example 3.4.5. The extended Vasil'ev code constructed by adding one more overall parity bit to the generalized Vasil'ev construction described in Theorem 3.4.2 is a $(39, 32, 4)$ partially robust code with $Q_{mc} = 0.5$ and $\omega_d = 16$.*

2. Alternatively let $a = 6$ and V be a $(31, 26, 3)$ perfect Hamming code. In this case we can construct a $(39, 32, 4)$ partially robust code with $Q_{mc} = 0.5$ and $\omega_d = 6$ at the price of larger hardware overhead for the encoder.

3. *For applications where hardware overhead is more critical, we can select* $a = 18$ *and* V *to be a* $(19, 14, 3)$ *shortened Hamming code. The resulting partially robust code will have* $\omega_d = 18$*, which is the biggest of the 3 discussed variants. However, the hardware overhead for the encoder of this implementation will be the smallest.*

3.4.2 Phelps Codes and Their Generalizations

We next review the basic construction of **Phelps Codes** that was proposed in (Phelps, 1983), analyze its detection kernel and conditional error masking properties and generalize the construction.

Theorem 3.4.3 ***(Phelps Code (Phelps, 1983))*** *Let* C, B *be two perfect linear Hamming codes of length* $n = 2^m - 1$*. Let* $\{C = C_0, C_1, C_2, \cdots, C_n\}$*,* $\{B = B_0, B_1, B_2, \cdots, B_n\}$ *be two partitions of* $GF(2^n)$ *with* $|C| = |C_i| = |B| = |B_i| = 2^{n-m}$ *such that the minimum distance between any two vectors in the same coset* C_i *(*B_i*) is three. Let* α *be any permutation of* $GF(2^m)$ *such that* $\alpha(\mathbf{0}) = \mathbf{0}$*. Represent each coset* C_i *(*B_i*) by a* m*-bit vector which is called the* ***coset vector*** *of* C_i *(*B_i*). Denote by* $[x] \in GF(2^m)$ *the coset vector of the coset that* x *belongs to. The coset vector can always be selected in such a way that* $[x] \oplus [y] = [x \oplus y]$*,* $x, y \in GF(2^n)$*. Define the* ***extended Phelps code*** C' *as follows:*

$$(c, p(c), b, p(b)) \in C' \text{ if and only if } \alpha([c]) = [b],$$

where $p : GF(2^n) \rightarrow GF(2)$ *is the linear parity function. Deleting any coordinate of* C' *will result in a perfect nonlinear Hamming code with* $d = 3$ *and length* $2^{m+1} - 1$ *which is called* ***Phelps code****.*

Corollary 3.4.2 *Let* P_α *be the nonlinearity of* α*. Phelps code is a*

$$(2^{m+1} - 1, 2^{m+1} - m - 2, 3)$$

perfect nonlinear Hamming code with $Q_{mc} = P_\alpha$ *and* $\omega_d = 2^{m+1} - 2m - 2$*, which is optimum with respect to bound (2.14). The extended Phelps code has* $d = 4$ *and the same* Q_{mc} *and* ω_d *as the Phelps code.*

Proof We first analyze the error masking properties for the extended Phelps code C' constructed in Theorem 3.4.3. The codewords of C' can be written in the following format:

$$c = (x_1, x_2, x_3, x_4),$$

where $x_1, x_3 \in GF(2^n), x_2 = p(x_1) \in GF(2), x_4 = p(x_3) \in GF(2)$ and $[x_3] = \alpha([x_1])$. Denote by $e = (e_1, e_2, e_3, e_4)$ the error vector, $e_1 \in GF(2^n)$, $e_2 \in GF(2)$, $e_3 \in GF(2^n)$, $e_4 \in GF(2)$. If $e_2 \neq p(e_1)$ or $e_4 \neq p(e_3)$, the error will never be masked. If $e_2 = p(e_1)$ and $e_4 = p(e_3)$, the error e will be masked by a codeword $c = (x_1, x_2, x_3, x_4)$ if and only if

$$[x_3] = \alpha([x_1]);$$
$$[x_3 \oplus e_3] = \alpha([x_1 \oplus e_1]).$$

The errors can be further divided into the following classes.

1. $e_1 \in C, e_3 \in B$. Then x_1 and $x_1 \oplus e_1$ belong to the same coset. x_3 and $x_3 \oplus e_3$ belong to the same coset. In this case the above two equations are always satisfied. Errors in this class form the detection kernel of C' and will be masked by all codewords. The number of these errors is $2^{2^{m+1}-2m-2}$.

2. $e_1 \in C$, $e_3 \notin B$. In this case x_1 and $x_1 \oplus e_1$ belong to the same coset. x_3 and $x_3 \oplus e_3$ belong to different cosets. Errors in this class will never be masked.

3. $e_1 \notin C$, $e_3 \in B$. Similar to the last case, errors in this class will never be masked.

4. $e_1 \notin C$, $e_3 \notin B$. The error masking equations are equivalent to:

$$\alpha([x_1] \oplus [e_1]) \oplus \alpha[x_1] = [e_3]. \tag{3.14}$$

α is a nonlinear function from $GF(2^m)$ to $GF(2^m)$ with nonlinearity P_α. The number of $[x_1]$ satisfying the above equation is no more than $P_\alpha \times 2^m$. There are at most $P_\alpha \times 2^{2n-m}$ codewords that mask the error $e = (e_1, e_2, e_3, e_4)$. Thereby the error will be masked by a probability no larger than P_α.

Without loss of generality, we assume that the parity check bit x_4 is deleted for all codewords. The resulting code is a perfect nonlinear Hamming code with minimum distance 3. It is easy to show that this code has the same Q_{mc} and ω_d as C' and is optimum with respect to bound (2.14). ■

Remark 3.4.1 *1. Codes C and B do not have to be linear Hamming codes. Generally speaking, ω_d of the Phelps code is equal to the sum of the dimension of K_d for C and B.*

2. To optimize Q_{mc}, P_α should be as small as possible. The best nonlinear function with the smallest P_α from $GF(2^m)$ to $GF(2^m)$ has $P_\alpha = 2^{-m+1}$ and is called almost perfect nonlinear function (Carlet and Ding, 2004). One example is x^3 (Maxwell, 2005), which is based on the cube operation in $GF(2^m)$.

Example 3.4.2 *Let $C = B$ be a $(7, 4, 3)$ perfect linear Hamming code defined by the parity check matrix*

$$H = \begin{pmatrix} 1001011 \\ 0101110 \\ 0010111 \end{pmatrix}.$$

$GF(2^7)$ can be partitioned into 8 cosets of the perfect Hamming code as it is shown in Table 3.1. The coset vectors $[x] \in GF(2^3)$ are assigned in such a way that $[x] \oplus [y] = [x \oplus y], x, y \in GF(2^7)$. For example, suppose $x = 0000001$ and $y = 0010000$. We have $[x] = 001, [y] = 111$ and $[x \oplus y] = [0010001]$. Since $H(x \oplus y) = 100 = h_1$, where h_1 is the first column of H, $x \oplus y \in C_7$ and $[x \oplus y] = 110 = [x] \oplus [y]$.

Let $\alpha([x]) = [x]^3$ with $P_\alpha = 2^{-r+1} = \frac{1}{4}$ (Example 2.5.1). According to Corollary 3.4.2, the resulting Phelps code has $\omega_d = 2^{3+1} - 2 \times 3 - 2 = 8$ and $Q_{mc} = P_\alpha = \frac{1}{4}$.

Theorem 3.4.3 can be further generalized to generate partially robust codes with Hamming distance three and four for any given length.

Table 3.1: Cosets of the $(7, 4, 3)$ linear Hamming code

	Coset Leader x	Coset Vector $[x]$
$C_0 = C$ $(B_0 = B)$	0000000	000
C_1 (B_1)	0000001	001
C_2 (B_2)	0000010	010
C_3 (B_3)	0000100	100
C_4 (B_4)	0001000	101
C_5 (B_5)	0010000	111
C_6 (B_6)	0100000	011
C_7 (B_7)	1000000	110

Theorem 3.4.4 ***(Generalized Phelps Code)*** *Let C be a $(n_1, k_1, 3)$ linear code and B be a $(n_2, k_2, 3)$ linear code. Without loss of generality, assume that $r_2 = n_2 - k_2 \geq r_1 = n_1 - k_1$. Let $\{C_0 = C, C_1, \cdots, C_{2^{r_1}-1}\}$ be a partition of $GF(2^{n_1})$ and $\{B_0 = B, B_1, \cdots, B_{2^{r_2}-1}\}$ be a partition of $GF(2^{n_2})$, where $|C_i| = 2^{k_1}, 0 \leq i \leq 2^{r_1} - 1$ and $|B_i| = 2^{k_2}, 0 \leq i \leq 2^{r_2} - 1$. Represent each coset C_i by a r_1-bit coset vector and each coset B_i by a r_2-bit coset vector. The coset vector can always be selected in such a way that $[x] \oplus [y] = [x \oplus y]$, where x, y are vectors in the same field and $[x]$ is the coset vector of the coset that x belongs to. Let α be any permutation of $GF(2^{r_1})$ such that $\alpha(\mathbf{0}) = \mathbf{0}$. Define C' as*

$$(c, p(c), b, p(b)) \in C' \text{ if and only if } (\mathbf{0}, \alpha([c])) = [b],$$

where $c \in GF(2^{n_1})$, $b \in GF(2^{n_2})$ and p is the linear parity function. C' is a $(n_1 + n_2 + 2, n_1 + k_2, 4)$ partially robust code with $Q_{mc} = P_\alpha$ and $\omega_d = k_1 + k_2$. Deleting any coordinate from every codeword of C' will result in a partially robust code with minimum distance three and the same value of Q_{mc} and ω_d as C'.

Theorem 3.4.4 can be proved in a similar way to Theorem 3.4.3 and Corollary 3.4.2.

Example 3.4.3 *Let C be a $(15, 10, 3)$ linear code whose parity check matrix is:*

$$H = \begin{pmatrix} 100001001011001 \\ 010000100101100 \\ 001001011001111 \\ 000100101100111 \\ 000010010110011 \end{pmatrix}.$$

Let B be a $(22, 17, 3)$ shortened Hamming code whose parity check matrix is:

$$H = \begin{pmatrix} 1000010010110011111000 \\ 0100001001011001111100 \\ 0010010110011111000110 \\ 0001001011001111100011 \\ 0000100101100111110001 \end{pmatrix}.$$

Let $\alpha([x]) = [x]^3$ with $P_\alpha = \frac{1}{16}$, where $[x] \in GF(2^5)$. Construct C' as described in Theorem 3.4.4. Then C' is a $(39, 32, 4)$ partially robust code with $\omega_d = 27$ and $Q_{mc} = P_\alpha = \frac{1}{16}$.

3.4.3 One Switching Constructions and Their Generalizations

Vasil'ev codes and Phelps codes usually have $\omega_d > 1$. Another important construction is **the switching construction**, which can generate nonlinear codes with $\omega_d = 1$. We first review the basic switching constructions presented by T. Etzion and A. Vardy in (Etzion and Vardy, 1994).

Theorem 3.4.5 ***(The Switching Code (Etzion and Vardy, 1994))*** *Let C be a perfect linear Hamming code of length $n = 2^r - 1$. Refer to the codewords of Hamming weight three as **triples**. Denote by T_i the subspace spanned by the triples that have one at the i_{th} bit. Let $\{C \oplus e_i^*\}$ be a translate of C, where $e_i^* \in GF(2^n)$ and has one at position i and zero elsewhere. Assume $\{T_i \oplus z\} \in C$ for some $z \in C$. A **switch** is a process of replacing the coset $\{T_i \oplus z\}$ with the coset $\{T_i \oplus z \oplus e_i^*\}$. The resulting one switching code C' defined by*

$$C' = \{C \backslash (T_i \oplus z)\} \bigcup \{T_i \oplus z \oplus e_i^*\} \tag{3.15}$$

for some $i \in \{1, 2, \cdots, n\}$ is a perfect nonlinear Hamming code of length $n = 2^r - 1$.

Corollary 3.4.3 *The one switching code C' is a partially robust code with $\omega_d = 2^{r-1} - 1$. In addition, $2^{2^r-1-r} - 2^{2^{r-1}-1}$ errors are masked with probability $1 - 2^{-2^{r-1}+1+r}$ and $2^{2^r-1-r} - 2^{2^{r-1}-1}$ errors are masked with probability $2^{-2^{r-1}+1+r}$. The code is optimum with respect to bound (2.14).*

Proof Without loss of generality, we assume that $z = 0$. C' is constructed by

replacing T_i with the coset $\{T_i \oplus e_i^*\}$. The errors can be divided into four classes as stated below.

1. $e \in T_i$. Since T_i is a linear subspace, we have

$$c \oplus e \in \{T_i \oplus e_i^*\} \text{ iff } c \in \{T_i \oplus e_i^*\};$$
$$c \oplus e \in \{C \backslash T_i\} \text{ iff } c \in \{C \backslash T_i\}.$$

   Hence $c \oplus e \in C'$ for every $c \in C'$. Errors in this class form the detection kernel of the code and will always be masked. The dimension of T_i is $\frac{n-1}{2}$(Phelps and Levan, 1995). Thereby the number of errors in this class is $2^{\frac{n-1}{2}} = 2^{2^{r-1}-1}$.

2. $e \in \{C \backslash T_i\}$. If $c \in \{T_i \oplus e\}$, $c \oplus e \in T_i$ and the error will be detected. If $c \in \{T_i \oplus e_i^*\}$, $c \oplus e \in \{T_i \oplus e_i^* \oplus e\}$, again the error will be detected. All the other codewords $c \in C', c \notin \{T_i \oplus e\}, c \notin \{T_i \oplus e_i^*\}$ will mask the error. Thereby errors in this class will be masked by a probability $\frac{|C|-2|T_i|}{|C|} = 1 - 2^{-2^{r-1}+1+r}$. The number of errors in this class is $|C| - |T_i|$.

3. $e \in \{\{C \backslash T_i\} \oplus e_i^*\}$. If $c \in \{T_i \oplus e_i^*\}$, $c \oplus e \in \{C \backslash T_i\}$ and the error will be masked. If $c \in \{T_i \oplus e \oplus e_i^*\}$, $c \oplus e \in \{T_i \oplus e_i^*\}$. The error will be masked. For all the other codewords the error will be detected. Thereby errors in this class will be masked by a probability $\frac{2|T_i|}{|C|} = 2^{-2^{r-1}+1+r}$. The number of errors in this class is $|C| - |T_i|$.

4. Errors that do not belong to the above three cases will always be detected. ■

Example 3.4.4 *Let C be a $(15, 11, 3)$ perfect linear Hamming code with the following parity check matrix*

$$H = \begin{pmatrix} 100010011010111 \\ 010011010111100 \\ 001001101011110 \\ 000100110101111 \end{pmatrix}.$$

There are $\frac{n-1}{2} = 7$ *codewords of Hamming weight three with the first bit equal to one, which are*

$$\begin{array}{l} 100100000000001, \\ 100000000001100, \\ 101000001000000, \\ 100000010100000, \\ 100000100000010, \\ 100001000010000, \\ 110010000000000. \end{array}$$

T_1 *is the subspace spanned by the above* 7 *codewords. The dimension of* T_1 *is seven. Construct* C' *by replacing* T_1 *with* $\{T_1 \oplus e_1^*\}$*, where* e_1^* *has one at the first bit and zero elsewhere.* C' *is a prefect nonlinear Hamming code with* $\omega_d = 7$*. There are* $2^{11} - 2^7$ *errors that are masked by probability* $\frac{7}{8}$ *and* $2^{11} - 2^7$ *errors that are masked by probability* $\frac{1}{8}$*.*

Another generalization of Theorem 3.4.5 was shown in (Phelps and Levan, 1995), which indicated that perfect nonlinear Hamming codes with $\omega_d = 1$ could be constructed by switching linear Hamming codes for multiple times.

Theorem 3.4.6 *(Phelps and Levan, 1995) Let* C *be a linear Hamming code of length* $n = 2^r - 1, r \geq 4$*, there exists* $z_i, 1 \leq i \leq r$ *such that the code*

$$C' = \{C \backslash (\bigcup_{i=1}^{k} T_i \oplus z_i)\} \bigcup \{\bigcup_{i=1}^{k} T_i \oplus z_i \oplus e_i^*\} \tag{3.16}$$

is a perfect nonlinear code with $\omega_d = 1$*.*

To end the section we summarize the optimality of different robust and partially robust codes with respect to bound (2.14) in Table 3.2 for the case when $r(d, n)$ is derived from the Hamming bound. The best candidates for protecting memories in channels with high MBU rate or laziness are generalized Vasil'ev codes and generalized Phelps codes. One switching code is better than linear Hamming code but worse than the above two candidates due to its larger Q_{mc}. We note that the multiple switching construction (Theorem 3.4.6) can generate robust codes with $\omega_d = 1$. However, its high encoding and decoding complexities and high Q_{mc} make the code not a viable alternative for memory protection.

3.4.4 Nonlinear Multi-Error Correcting Codes

The Hamming distance is 3 for all minimum distance partially robust codes presented in the last section. These codes can only be used to correct single bit errors. In this section, two constructions of nonlinear multi-error correcting codes will be described. The codes presented in this section can correct not only multi-bit errors but also multi-digit errors in $GF(p)$.

Multi-Error Correcting Codes Based on Concatenations

The first construction of nonlinear multi-error correcting codes is based on the idea of concatenating linear and nonlinear redundant digits.

Theorem 3.4.7 *Let $f : GF(p^k) \rightarrow GF(p^{r_n})$ be a nonlinear function with nonlinearity P_f. Let $V = \{(z, \phi(z))\}$ be a linear systematic code with Hamming distance d, where $z \in GF(p^{k+r_n})$ and $\phi(z) : GF(p^{k+r_n}) \rightarrow GF(p^{r_l})$ is the encoding function. The code defined by*

$$\{(y, f(y), \phi(z))\}, \tag{3.17}$$

where $y \in GF(p^k), f(y) \in GF(p^{r_n}), z = (y, f(y)) \in GF(p^{k+r_n})$ and $\phi(z) \in GF(p^{r_l})$ is a robust error correcting code with Hamming distance d and $K_d = \{0\}$. Any nonzero error will be detected with a probability of at least $1 - P_f$.

Proof Let $e = (e_1, e_2, e_3)$ be the error vector, where $e_1 \in GF(p^k), e_2 \in GF(p^{r_n})$ and $e_3 \in GF(p^{r_l})$. The error masking equations can be written as:

$$f(y \oplus e_1) = f(y) \oplus e_2; \tag{3.18}$$

$$\phi(z + (e_1, e_2)) = \phi(z) + e_3. \tag{3.19}$$

1. If $e_1 = 0$ and e_2, e_3 are not both 0, at least one of the equations shown above will not be satisfied. The error will always be detected.

2. If $e_1 \neq 0$, from the definition of nonlinear functions, there are at most $P_f p^k$

solutions of y for (3.18). Thereby, the error will be masked with a probability of at most P_f. ■

The resulting code C in Theorem 3.4.7 has the same Hamming distance d and the same error correcting capability t as the linear systematic code V.

Example 3.4.5 ***(Shortened BCH Code)*** *Let C be a $(44, 32, 5)$ shortened BCH code, where $x \in GF(2^{32})$. Let $f : GF(2^{32}) \rightarrow GF(2)$ be a perfect nonlinear function defined by $f(x = (x_1, x_2, ..., x_{32})) = x_1 \cdot x_2 \oplus x_3 \cdot x_4 \oplus ... \oplus x_{31} \cdot x_{32}$ (Karpovsky et al., 2008b). Then the code $C' = \{(x, Px, f(x))\}$ is a robust code with minimum distance 5. For this code, $\omega_d = 0$, $Q(e) = 0$ when $||e|| < 5$ and $Q(e) \leq 0.5$ when $||e|| \geq 5$.*

Multi-Error Correcting Generalized Vasil'ev Codes

In Section 3.4.1, we generalized Vasil'ev constructions to generate Hamming codes with distance 3 and arbitrary length n. The presented codes required the same number of redundant digits as linear (shortened) Hamming codes. In this section, Vasil'ev constructions will be further generalized to construct p-ary codes with any given distance d and dimension k.

Theorem 3.4.8 *Let $q_1 = p^{l_1}$ and $q_2 = p^{l_2}$, $l_1 \geq l_2 \geq 1$. Let V be a (n_1, k_1, d) q_1-ary code and $U = \{(u, uP)\}$ be a (n_2, k_2, d') q_2-ary code, where $u \in GF(q_2^{k_2})$, $k_2 \leq n_1$, $r_2 = n_2 - k_2$, $d' \geq d - 1$ and P is a $k_2 \times r_2$ encoding matrix in $GF(q_2)$ (the last r_2 columns of the generator matrix of the code in standard form). Let $f : GF(q_1^{k_1}) \rightarrow GF(q_2^{r_2})$ be an arbitrary mapping such that $f(\mathbf{0}), \mathbf{0} \in GF(q_1^{k_1})$ is equal to zero and $f(y) \oplus f(y') \neq f(y \oplus y')$ for some $y, y' \in GF(q_1^{k_1})$, where $\oplus$ is the digit-wise addition in $GF(p)$. Let $u = (u_1, u_2, \cdots u_{k_2})$, where $u_i \in GF(q_2)$. Let $\beta(u) = ((u_1, \mathbf{0}), (u_2, \mathbf{0}), \cdots, (u_{k_2}, \mathbf{0}))$, where $\mathbf{0} \in GF(p^{l_1 - l_2}), (u_i, \mathbf{0}) \in GF(q_1)$ and $\beta(u) \in GF(q_1^{k_2})$. Denote by v_k the information bits of $v \in V$. The code defined by*

$$C = \{(u, (\beta(u), \mathbf{0}) \oplus v, uP \oplus f(v_k))\}, \mathbf{0} \in GF(q_1^{n_1 - k_2}) \tag{3.20}$$

is a $(l_1 n_1 + l_2 n_2, l_1 k_1 + l_2 k_2, d)$ p-ary partially robust code with $|K_d| = p^{l_2 k_2}$. The remaining errors are detected with a probability of at least $1 - P_f$, where P_f is the

nonlinearity of f. Consider elements of U in $GF(q_2)$ and elements of V in $GF(q_1)$ as equivalent ***digits***. *The codeword of C has $n_1 + n_2$ digits and C has the same digit error correcting capability as V.*

Proof Let $c = (u, (\beta(u), \mathbf{0}) \oplus v, uP \oplus f(v_k)), c' = (u', (\beta(u'), \mathbf{0}) \oplus v', u'P \oplus f(v'_k))$ be two codewords of C. The Hamming distance between c and c' is

$$\begin{aligned} ||c \oplus c'|| &= ||u \oplus u'|| + ||(\beta(u), \mathbf{0}) \oplus v \oplus (\beta(u'), \mathbf{0}) \oplus v'|| \\ &+ ||uP \oplus f(v_k) \oplus u'P \oplus f(v'_k)|| \\ &\geq ||v \oplus v'||. \end{aligned}$$

1. If $v \neq v'$, $||c \oplus c'|| \geq d$ because the Hamming distance of V is d.
2. If $v = v'$, $||c \oplus c'|| = 2 \times ||u \oplus u'|| + ||uP \oplus uP'|| \geq d$ because the Hamming distance of $U = \{(u, uP)\}$ is at least $d - 1$.

Thereby, the Hamming distance of C is d. We say that an error e is masked by a codeword c if $e \oplus c = c' \in C$. Let H be the parity check matrix of V. An error $e = (e_1, e_2, e_3)$ where $e_1 \in GF(q_2^{k_2}), e_2 \in GF(q_1^{n_1})$ and $e_3 \in GF(q_2^{r_2})$ is masked if and only if $H((\beta(e_1), \mathbf{0}) \oplus e_2)$ is zero and $f(\tilde{v}_k) \oplus f(v_k) \oplus e_1P \oplus e_3 = 0$, where $\tilde{v}_k$ is the information part of $\tilde{v} = v \oplus (\beta(e_1), \mathbf{0}) \oplus e_2$ and $\mathbf{0} \in GF(q_1^{n_1 - k_2})$. The errors can be divided into four classes as follows.

1. $(\beta(e_1), \mathbf{0}) = e_2$ and $e_1P = e_3$. The error will always be masked. The number of errors in this class is $q_2^{k_2}$;
2. $(\beta(e_1), \mathbf{0}) = e_2$ but $e_1P \neq e_3$. The error will always be detected. There are $q_2^{n_2} - q_2^{k_2}$ errors belonging to this class;
3. $H((\beta(e_1), \mathbf{0}) \oplus e_2)$ is zero but $(\beta(e)_1, \mathbf{0}) \neq e_2$. The error masking probability depends on the nonlinear function f. In the worst case, a specific error will be masked by $P_f \times |C|$ codewords. The number of errors in this class is $q_2^{n_2}(q_1^{k_1} - 1)$;

4. $H((\beta(e_1), \mathbf{0}) \oplus e_2)$ is not zero. The error will always be detected. The number of errors is $q_2^{n_2}(q_1^{n_1} - q_1^{k_1})$. ■

Remark 3.4.2 *Binary Vasil'ev code presented in (Vasil'ev, 1962) is a special case where $q_1 = q_2 = 2$, $\{(u, Pu)\}$ is a linear parity code with minimum distance two and V is a perfect Hamming code.*

Some nonlinear multi-error correcting codes as good as linear codes in terms of the number of redundant digits for the same distance and length can be generated based on the above construction.

Example 3.4.6 *In (MacWilliams and Sloane, 1998), it was shown that the largest possible k for binary codes with $n = 63$ and $d = 5$ is 52. Let V be a $(63, 52, 5)$ binary code ($q_1 = 2$). Let $\{(u, uP)\}$ be a $(4, 1, 4)$ binary repetition code that contains only 2 codewords 0000 and 1111 ($q_2 = 2$). Select f to be a quadratic perfect nonlinear function $f = s_1 \bullet s_2 \oplus s_3 \bullet s_4 \oplus \cdots \oplus s_{13} \bullet s_{14}$ with $P_f = \frac{1}{8}$ where $s_i \in GF(2^3)$ and $\bullet$ is the multiplication in $GF(2^3)$. A $(67, 53, 5)$ partially robust nonlinear BCH code can be constructed as described in Theorem 3.4.8. This code has Hamming distance five and only one undetectable nonzero error and has the same number of redundant bits as linear BCH codes. All the other errors are detectable with a probability of at least* 0.875.

Theorem 3.4.8 can also generate nonbinary multi-error correcting codes that are as good as linear codes in terms of the number of redundant digits.

Example 3.4.7 *In general, a nonbinary BCH code with Hamming distance d in $GF(q)$ has length $n = q^m - 1$ and dimension $k = q^m - 1 - (d-1)m$. Let $n = 49$, $q = 7$ and $d = 5$. The corresponding nonbinary BCH code is constructed by shortening the BCH code with $m = 3$ and $k = 7^3 - 13$. The number of redundant digits is 12. Let V be a $(48, 40, 5)$ nonbinary BCH code in $GF(7)$ ($m = 2$), U be a $(5, 1, 5)$ repetition code in $GF(7)$ and f be a perfect nonlinear function from $GF(7^{40})$ to $GF(7^4)$. The resulting nonlinear nonbinary BCH code constructed as in Theorem 3.4.8 has the same number of redundant digits as the linear BCH codes in $GF(7)$. Only 7 errors are undetectable (including the all-zero error vector). All other errors are detected with a probability of at least $1 - 7^{-4}$.*

3.5 Summary

Robust codes can be constructed based on nonlinear functions. The worst case error masking probability of robust codes is determined by the nonlinearity of the encoding function. The best robust codes have no undetectable errors and minimize the worst case error masking probability among all codes with the same parameters. Perfect robust codes exist for very limited parameters and can only be non-systematic. Systematic robust codes cannot be perfect. The best systematic codes are called optimum systematic codes. The encoding functions of these codes are perfect nonlinear functions.

Robust codes usually have high encoding and decoding complexity. To achieve a tradeoff between the robustness and the overhead, partially robust codes are proposed in (Karpovsky and Taubin, 2004). Partially robust codes still have undetectable errors. But the number of these errors is largely reduced compared to linear codes. Robust and partially robust codes have been used for the protection of AES (Karpovsky et al., 2004; Kulikowski et al., 2008b), the build of reliable memories (Karpovsky et al., 2007), the error detection in lazy channels (Karpovsky et al., 2008a) and the check point verification (Karpovsky et al., 2008a).

To provide a fully protection against errors with small multiplicities and meanwhile maintain nearly equal protection against other errors, minimum distance robust and partially robust codes, e.g. generalized Vasil'ev codes, generalized Phelps codes, etc, are presented. These codes can be as good as the best known linear error detecting codes in terms of the Hamming distance and has much less undetectable errors. Minimum distance robust and partially robust codes with a Hamming distance at least 3 can be used for error correcting. Nonlinear single-bit error correcting codes were used in (Wang et al., 2009c) for the protection of SRAMs. Nonlinear multi-error correcting codes were used in (Wang et al., 2010b) to build reliable MLC NAND

Flash memories.

Constructions of robust and partially robust codes were co-developed with Konrad Kulikowski. Constructions of minimum distance robust and partially robust codes generalized from Vasil'ev codes, Phelps codes, etc were proposed by the author of the dissertation in (Wang et al., 2009c; Wang et al., 2010a; Wang et al., 2010b)

Table 3.2: Optimality of robust and partially robust codes with respect to bound (2.14) ($r(d,n)$ is derived from the Hamming bound)

	n	k	ω_d	Q_{mc}	d	r_N	Perfect	Optimum
Vasil'ev Codes (Theorem 3.4.1)	2^r-1	2^r-1-r	$2^{r-1}-1$	0.5	3	1	$\surd$	-
Generalized Vasil'ev Code (Theorem 3.4.2)	$a+m+1$	$a+k_V$	a	0.5	3	1	-	$\surd$
Phelps Code (Theorem 3.4.3)	2^r-1	2^r-1-r	2^r-2r	P_α	3	$r-1$	-	$\surd$
Generalized Extended Phelps Code (Example 3.4.3)	39	32	28	$\frac{1}{4}$	4	6	-	-
One Switching Code (Theorem 3.4.5)	2^r-1	2^r-1-r	$2^{r-1}-1$	$1-2^{-2^{r-1}+1+r}$	3	1	-	$\surd$
$(x,(Px)^3)$ (Karpovsky and Taubin, 2004)	$k+r$	k	$k-r$	2^{-r+1}	1, 2[1]	r	-	-
Quadratic Systematic Code[2] (Karpovsky et al., 2007)	$(2s+1)r$	$2sr$	0	2^{-r}	1	r	-	$\surd$
Robust Hamming Code[3] (Kulikowski et al., 2008b)	2^r	2^r-1-r	0	0.5	3	1	-	$\surd$

[1]: The distance of the code depends on r.

[2]: The codeword of the quadratic systematic code is in the format of $(x_1, x_2, \cdots x_{2s}, x_{2s+1})$, where $x_i \in GF(2^r), 1 \leq i \leq 2s+1$ and $x_{2s+1} = x_1 \bullet x_2 \oplus x_3 \bullet x_4 \oplus \cdots \oplus x_{2s-1} \bullet x_{2s}$. $\bullet$ is the multiplication in $GF(2^r)$. (Karpovsky et al., 2008b)

[3]: The codeword of the robust Hamming code is in the format of $(x, Px, f(x))$, where (x, Px) is the codeword of a $(2^r-1, 2^r-1-r, 3$ perfect linear Hamming code and $f : GF(2^k) \to GF(2)$ is a nonlinear function. When f is a perfect nonlinear function, $Q_{mc} = 0.5$.

Chapter 4

Multilinear Codes

Generally speaking, the computation of nonlinear functions cannot be easily simplified. As a result, robust codes based on nonlinear functions usually have a high encoding and decoding complexity. Partially robust codes were proposed in (Karpovsky and Taubin, 2004) as an alternative to robust codes to achieve a tradeoff between the robustness and the hardware overhead. However, partially robust codes still require the computation of nonlinear functions. Secure cryptographic devices based on partially robust codes may still have a large hardware overhead. For example, in (Kulikowski et al., 2008b), it was shown that using partially robust codes to protect the linear portion of AES requires an area overhead of more than 300%.

In this Chapter, we present a novel error detection technique based on the idea of randomly selecting a code from multiple linear codes for each encoding and the corresponding decoding operation. The resulting codes are called multilinear codes. These codes have similar error detection capabilities to robust codes while requiring much less hardware overhead due to the fact that no nonlinear operations are needed for the encoder and decoder.

Algebraic multilinear codes have been used in (Wang and Karpovsky, 2010) for the build of robust FSMs for resilient to advanced fault injection attacks. Arithmetic multilinear codes have been used in (Wang et al., 2009b) for the design of secure multipliers.

4.1 Multilinear Algebraic Codes

For the remainder of this Chapter we denote by $\{C_i, 1 \leq i \leq l, l \geq 2\}$ the set of linear codes, where l is the number of different codes in the set. We first propose several methods of constructing linear algebraic codes $C_i, 2 \leq i \leq l$ from C_1 in such a way that randomly selecting $C_i, 1 \leq i \leq l$ have much less or even no undectable errors. For the randomization a standard low-rate true random number generator is used, which is available in most cryptographic devices.

4.1.1 Constructions Based on Swapping the Redundant Bits

Construction 4.1.1 *Let C_1 be a (n, k) linear code with Hamming distance larger than 2. $C_i, 2 \leq i \leq l, l = k$ is constructed by swapping the first and the i_{th} information bits of C_1. If we randomly select $C_i, 1 \leq i \leq l = k$ to encode the original messages with equal probability, the only undetectable error is the codeword of C_1 with all $1's$ in the information part. Errors that have the same value for the first $k-1$ information bits will be masked with probability $\frac{k-1}{k}$, which is the maximum conditional error masking probability.*

Proof Undetectable errors are codewords that belong to all of the $l = k$ linear codes $C_1, C_2, \cdots, C_l$. Because the Hamming distance of C_1 is larger than 2, the intersection of these codes contains only the vector (also a codeword) with all $1's$ information bits and the vector with all $0's$ information bits. Errors which have the same value for the first $k-1$ information bits belong to $k-1$ linear codes. So it will be masked with probability $\frac{k-1}{k}$ if we select the codes with equal probability. Obviously, this is the maximum conditional error masking probability. ■

When implemented in hardware, the overhead of Construction 4.1.1 may be excessive. To reduce the hardware overhead, we can use only $l, 2 \leq l < k$ linear codes. Generally speaking, when we randomly select $l, 2 \leq l \leq k$ linear codes to encode the messages, the number of undetectable errors is 2^{k-l+1} (including the all $0's$ vector).

The maximum error masking probability for all conditionally detectable errors is $\frac{l-1}{l}$. The simplest case is to use only C_1 and C_2 to encode the messages, where C_2 is built by swapping the first and the second information bits of C_1. This method requires only 2 more $2:1$ multiplexer for the encoder while the number of undetectable errors is reduced by 50% compared with the method using only C_1. All conditionally detectable errors will be detected with probability 0.5.

Another variation of Construction 4.1.1 is to swap the redundant bits instead of the information bits of C_1. Suppose $C_i, 2 \leq i \leq r$ is constructed by swapping the first and the i_{th} redundant bits of C_1. Assume that all 2^r binary vectors are possible for the redundant part of C_1. If we randomly select $l, 2 \leq l \leq r$ linear codes to encode the original messages with equal probability, the number of undetectable errors is 2^{k-l+1}. The maximum error masking probability for conditionally detectable errors is $\frac{l-1}{l}$. For this variation the smallest possible number of undetectable errors is 2^{k-r+1} which is larger than that can be achieved by swapping information bits due to the fact that only r different codes can be constructed.

Example 4.1.1 *We compare the hardware complexity for the encoder and the number of undetectable errors for the architectures that utilize different numbers of linear codes constructed by swapping information bits of the original code. Let* C_1 *be a* $(39, 32)$ *Hsiao code (Hsiao, 1970) whose parity check matrix is in the standard form* $H = [I, P]$*, where* I *is a* 7×7 *identity matrix and* P *is a* 7×32 *predictor matrix defined as follows.* $C_i, 2 \leq i \leq 32$ *is constructed by swapping the first and the* i_{th} *information bits of* C_1

$$P = \begin{bmatrix} 11111111000000100001001010000011 \\ 00001001111111110010010010000100 \\ 00010000000100001111111100110110 \\ 00100010001001011000000011111111 \\ 01100101010010010000111101101000 \\ 10000110100011101111100000001000 \\ 11011000111100000100000101010001 \end{bmatrix}.$$

The hardware complexity for the encoder, the number of undetectable errors and the maximum error masking probability of conditionally detectable errors for four

Table 4.1: Hardware complexity for the encoder, number of undetectable errors and maximum conditional error masking probabilities for schemes using different number of codes from Construction 4.1.1 ($n = 39, k = 32$)

Number of Codes	Number of Gates	Number of Undetectable Errors	Maximum conditional error masking probability
1	70	2^{32}	-
2	77	2^{31}	0.5
4	96	2^{29}	0.75
8	153	2^{25}	0.875

schemes are shown in Table 4.1. The first column is the number of codes we randomly select to encode the messages. Row 1 *corresponds to the case when only* C_1 *is used.* 70 *2-input gates and inverters are required to build the encoder. When randomly select* 2 *linear codes, we only need* 7 *extra gates and the number of undetectable errors is reduced by* 50% *compared with the case when only a single linear code is used. Increasing the number of codes can further decrease the number of undetectable errors. However, this is at the cost of larger hardware overhead and worse conditional error masking probability.*

4.1.2 Constructions Based on Circular Shifts

Another simple way to construct $C_i, 2 \leq i \leq l$ from C_1 is to circularly shift the redundant bits of C_1 as outlined below.

Construction 4.1.2 *Let* C_1 *be a* (n, k) *linear code with* $r = n-k \leq k$ *redundant bits. Denote by* $H = [I \mid P]$ *the parity check matrix of* C_1*, where* I *is a* $r \times r$ *identity matrix and* P *is a* $r \times k$ *predictor matrix. Assume that the rank of* P *is* r*. Construct* C_2 *by circularly shifting the redundant part of* C_1 *by 1 bit. If we randomly select* C_1 *and* C_2 *to encode the original messages with equal probability, the number of undetectable errors is* 2^{k-r+1}*. In addition, there are* $2^{k+1} - 2^{k-r+2}$ *errors which will be detected with probability* 0.5*.*

Proof Denote by $x = (x_1, x_2, \cdots, x_n)$ the codeword of a (n, k) linear code and assume that the first r bits are redundant bits. If $(x_1, x_2, \cdots, x_r, x_{r+1}, \cdots, x_n)$ belongs to both C_1 and C_2, then from the construction method of C_2 we know that $(x_2, x_3, \cdots, x_r, x_1, x_{r+1}, \cdots, x_n)$ also belongs to C_2. Hence the sum $(x_1 \oplus x_2, x_2 \oplus x_3, \cdots, x_r \oplus$

$x_1, 0, \cdots, 0)$ is another codeword of C_2. Because the information part is all $0's$, the redundant part should also be all $0's$. Thereby $x_1 \oplus x_2 = 0, x_2 \oplus x_3 = 0, \cdots, x_r \oplus x_1 = 0$. So $x_1 = x_2 = \cdots = x_r \in \{0, 1\}$. Given the assumption that the rank of the predictor matrix P is r, all 2^r values are possible for the redundant part of the code. There are 2^{k-r} codewords that can generate each value of the redundant part. Hence the number of undetectable errors is equal to the size of the intersection of the two code which is 2^{k-r+1}. ■

Example 4.1.2 *$(x, (Px)^3)$ is a partially robust code, where $x \in GF(2^k)$, P is a $r \times k$ matrix in $GF(2)$, $Px \in GF(2^r)$ and $y^3(y = Px)$ is a cube operation in Galois Field $GF(2^r)$ (Karpovsky and Taubin, 2004). The number of undetectable errors of $(x, (Px)^3)$ code is 2^{k-r}. All conditionally detectable errors are masked with probability 2^{-r+1}. Compared with $(x, (Px)^3)$ code, Construction 4.1.2 has nearly the same number of undetectable errors but requires much less hardware overhead to implement. As an illustrative example, Table 4.2 compares the hardware overhead for the encoder, the number of undetectable errors as well as the maximum conditional error masking probability for these two codes when $n = 39, k = 32$. P is selected to be the same matrix as in Example 4.1.1. Only 95 2-input gates and inverters are required for the encoder of circularly shifting method while $(x, (Px))$ needs 514. The gap will become even larger when r increases.*

Table 4.2: Comparison of $(x, (Px)^3)$ code and Construction 4.1.2

Codes	Number of Gates	Number of Undetectable Errors	Maximum conditional error masking probability
$(x, (Px)^3)$	514	2^{25}	2^{-6}
Construction 4.1.2	95	2^{26}	0.5

Construction 4.1.2 can be further improved to reduce the maximum conditional error masking probability. Let C_1 be a (n, k) linear code. Assume that we can find m numbers $s_i, 1 \leq i \leq m < r = n-k$, such that $s_i, 1 \leq i \leq m$ and r are mutually prime. $C_i, 2 \leq i \leq m+1$ is constructed by circularly shifting the redundant part of C_1 by s_{i-1} bits. If we randomly select $C_i, 1 \leq i \leq m+1$ to encode the original messages with

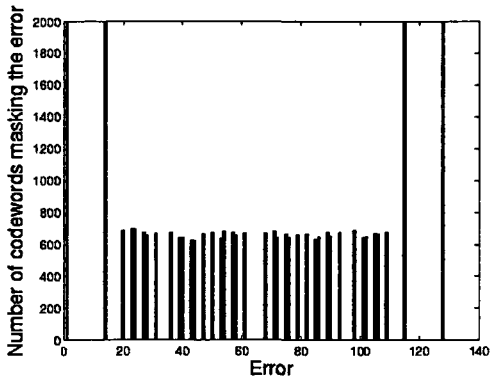


Figure 4·1: Error detection properties of circularly shifting and randomly selecting $(7,4)$ Hamming code

equal probability, the number of undetectable errors is 2^{k-r+1}, assuming the rank of the predictor matrix of C_1 is r. In addition, $(m+1)\cdot(2^k - 2^{k-r+1})$ errors will be detected with probability $\frac{1}{m+1}$.

When r is prime, s_i can be any integer in the range of $[1, r-1]$. In this case the maximum conditional error masking probability is $\frac{1}{r}$.

Example 4.1.3 *Let C_1 be a $(7,4)$ linear perfect Hamming code. $r = n - k = 3$ is prime. Construct $C_i, 2 \leq i \leq 3$ by circularly shifting the redundant part of C_1 by $i-1$ bits. Figure 4·1 shows the error masking properties for all $2^7 - 1$ nonzero errors. For each error we encode 2000 randomly selected messages. A C_i is randomly chosen to encode each of the message. As we can see from Figure 4·1, $2^{k-r+1} = 4$ errors are undetectable (including the all 0's vector). All the conditionally detectable errors are masked by about 660 codewords, which means they are masked with probability $\frac{1}{r} = \frac{1}{3}$.*

The more codes we randomly select from, the better the error detection ability we can achieve. Randomly selecting from more codes can result in either a decrease of the number of undetectable errors (Example 4.1.1) or a smaller maximum conditional error masking probability (Example 4.1.3). On the other hand, randomly selecting more linear codes means more complicated encoding and decoding strategies which may result in larger hardware overhead. An apparent question is how to randomly

select codes to optimize either the number of undetectable errors or the maximum conditional error masking probability. If we randomly select from l different linear codes, the best conditional error masking probability is $\frac{1}{l}$ which can be achieved when there are no errors belonging to more than one code except for the undetectable errors. The conditions for minimizing the number of undetectable errors when randomly selecting $l \leq \lfloor \frac{k}{r} \rfloor + 1$ linear codes is derived in the next section.

4.1.3 Randomly Selecting from Non-Overlapping Linear Codes

It is easy to show that the smallest possible dimension of the intersection of $l \leq \lfloor \frac{k}{r} \rfloor + 1$ different (n, k) linear codes is $k-(l-1)r$, where $r = n-k$ is the number of redundant bits. For linear codes, every single redundant bit can be written as a seperate function of the information bits. Denote by $f_{i,j}$ the encoding function for the i_{th} code to generate the j_{th} redundant bit, where $1 \leq i \leq l, 1 \leq j \leq r$. The errors belonging to the intersection of all l linear codes should satisfy the following equations: $f_{1,j} = f_{2,j} = \cdots = f_{l,j}$ or equivalently $f_{1,j} \oplus f_{2,j} = 0, f_{1,j} \oplus f_{3,j} = 0 \cdots f_{1,j} \oplus f_{l,j} = 0$ where $j = 1, 2, \ldots, r$. There are in total $(l-1)r$ equations and k unknowns (information bits of the code). The smallest possible dimension of the intersection is $k-(l-1)r$ which can be achieved when all these $(l-1)r$ equations are linearly independent. We next give a construction which can optimize the number of undetectable errors for any given $l \leq \lfloor \frac{k}{r} \rfloor + 1$.

Construction 4.1.3 *Suppose we want to construct $l \leq \lfloor \frac{k}{r} \rfloor + 1$ linear systematic codes such that the dimension of the intersection of these codes is minimum. Denote by H_i the parity check matrix of the i_{th} linear code. Without loss of generality, assume that the first r bits of any codeword are the redundant bits and the parity check matrices are in standard form $H_i = [I_r \mid P_i]$, where I_r is a $r \times r$ identity matrix and P_i is a*

$r \times k$ predictor matrix. Given P_1, P_i $2 \leq i \leq l$ can be constructed as follows.

$$\begin{aligned} P_2 &= P_1 \oplus [I_r, 0_{r,k-r}] \\ P_3 &= P_1 \oplus [0_{r,r}, I_r, 0_{r,k-2r}] \\ &\vdots \\ P_l &= P_1 \oplus [0_{r,(l-2)r}, I_r, 0_{r,(k-(l-1)r)}] \end{aligned}$$

where I_r is a $r \times r$ identity matrix and $0_{i,j}$ is a $i \times j$ all zero matrix.

Example 4.1.4 *In this example, we construct two $[10, 5]$ linear systematic codes such that the intersection of these two codes contains only the $0's$ vector. We select the first code C_1 to be a shortened Hamming code with the following parity check matrix.*

$$H_1 = \begin{bmatrix} 1000010101 \\ 0100011111 \\ 0010011011 \\ 0001000111 \\ 0000100101 \end{bmatrix}.$$

According to Construction 4.1.3, the parity check matrix of the second code can be computed as follows:

$$H_2 = \begin{bmatrix} 1000010101 \\ 0100011111 \\ 0010011011 \\ 0001000111 \\ 0000100101 \end{bmatrix} + \begin{bmatrix} 0000010000 \\ 0000001000 \\ 0000000100 \\ 0000000010 \\ 0000000001 \end{bmatrix}.$$

It is easy to verify that the dimension of the intersection is $k-r = 0$. The only vector belonging to both codes is the all $0's$ vector.

In Construction 4.1.3, $P_i, 2 \leq i \leq l$ is built by flipping r bits of r columns in the original predictor matrix P_1, one bit for each column. Generally speaking, this method cannot guarantee that all the linear codes have the same distance as the original code C_1. In the above example, the distance of C_2 is 2 instead of 3. However, by carefully selecting the parity check matrix for C_1 or adjusting the flipping positions, it is possible to make the other linear codes have the same distance as C_1. For instance,

we can construct another $(10, 5)$ linear code C_2^* with the parity check matrix computed as follows. The minimum distance of C_2^* becomes 3 in this case while the intersection of C_1 and C_2^* still contains only the all $0's$ vector.

$$H_2^* = \begin{bmatrix} 1000010101 \\ 0100011111 \\ 0010011011 \\ 0001000111 \\ 0000100101 \end{bmatrix} + \begin{bmatrix} 0000010000 \\ 0000000010 \\ 0000000100 \\ 0000001000 \\ 0000000001 \end{bmatrix}.$$

4.1.4 General Analysis of Fault Detecion Ability of Multilinear Codes

A big difference between linear codes and the proposed constructions based on randomly selecting multiple linear codes is that our method has conditionally detectable errors. The detection of errors is message dependent. If the error is masked by one codeword, it is still possible that it will be detected by a different code at the next moment. The longer the same error stays, the higher the detection probability is. Thereby in channels where errors tend to repeat themselves, our method has higher error detection ability than classical linear codes.

For applications utilizing cryptographic devices, we are more concerned about the fault detection ability of the code. The same fault may manifest itself as different error patterns at the output of the devices. When the same fault stays for t consecutive clock cycles, a general analysis of the fault detection abilities of the proposed method is shown in the next theorem.

Theorem 4.1.1 *Let $C_1, C_2 \cdots C_L$ be L different linear (n, k) codes. Assume that single or multiple faults stay for $t = aL + b, a \geq 0, 0 \leq b \leq L - 1$ consecutive clock cycles and may manifest themsevles as s different error patterns e_i, $1 \leq i \leq s \leq 2^n$ with probability $p(e_i)$ respectively.(We assume e_i may be the all $0's$ vector.) $P_j = \sum_{e_i \in C_j, 1 \leq i \leq s} p(e_i), 1 \leq j \leq L$ is the probability that faults manefest themselves as errors which are codewords of C_j. Denote by W_t the probability that faults are not detected after t clock cycles. If we circularly select $C_1, C_2 \cdots C_L$ to encode the message*

at every clock cycle, $W_t = \prod_{1 \le j \le L} P_j^{a+1-H(j-b-1)}$, *where* $H(j-b-1)$ *is the unit step function. If we randomly select the codes,* $W_t = (\frac{1}{L}\sum_{i=1}^{L} P_i)^t$.

Proof An error e is masked by a linear code C iff $e \in C$. W_t is equal to the probability that $e_m \in C_m, 1 \le m \le t$, where e_m is the error vector introduced by the faults and C_m is the selected code at the m_{th} clock cycle. Thereby $W_t = \prod_{1 \le m \le t} p(e_m \in C_m)$. ■

Circularly selecting codes at each clock cycle is not suitable for cryptography applications because the attackers can circumvent the protection schemes if he knows what codes are used at each clock cycle. The advantage of circularly selecting, however, is that every error staying for at least L consecutive clock cycles can be 100% detected if the intersection of the L codes contains only the all $0's$ vector. When $L = 2$, all nonzero errors can be detected after staying for at most two clock cycles as long as C_1 and C_2 are non-overlapping and this is very useful for applications related to many communication channels and some computational channels where errors tend to repeat themselves with high probability, e.g. linear computational network consisting of XOR gates only.

To demomstrate the advantage of the proposed method, we compare the fault masking probability after t consecutive clock cycles for three error protection schemes for linear networks. The first one is based on a single $(20, 15)$ shortened linear Hamming code. Denote it by C_1. The second one utilizes four $(20, 15)$ linear codes $C_i, 1 \le i \le 4$ whose intersection contains only the all $0's$ vector. For the construction of $C_i, 2 \le i \le 4$, please refer to Construction 4.1.3. The third method is based on the $(x, (Px)^3)$ partially robust code. We can select P to be the same predictor matrix as for C_1.

To simplify the analysis. We assume that a stuck at fault occurs in the linear network. The fault manifests itself as the same nonzero error e at the output of the

network with probability 0.5. If e is a codeword of C_1, after t clock cycles the error will be masked by the shortened Hamming code with probability 1. For method 2, we randomly select $C_i, 2 \leq i \leq 4$ with equal probability. If e belongs to the intersection of 3 codes, it will be masked with probability 0.875^t after t clock cycles according to Theorem 4.1.1. If e only belongs to one code, the error masking probability after t clock cycles is 0.625^t. The partially robust code $(x, (Px)^3)$ has $2^{k-r} = 2^{10}$ undetectable errors. If e is undetectable by $(x, (Px)^3)$, it will be masked with probability 1 regardless of t. If e is conditionally detectable by $(x, (Px)^3)$, the error masking probability after t clock cycles is $(0.5 + 0.5 \cdot 2^{-r+1})^t$.

Figure 4·2 plots the fault masking probabilities after 10 clock cycles for the three alternatives. As expected, when considering the worst case fault masking probabilities, linear code is much worse than the other two. The fault will be masked no matter how many clock cycles it stays if it manifests as a codeword of the linear code. The method based on multilinear codes is much better than that based on single linear code. The performance of multilinear codes also depends on how the fault manifests itself. The less codes the manifested error belongs to, the better the fault detection ability is. One disadvantage of $(x, (Px)^3)$ code is that it still has undetectalbe errors. Even if the manifested error is conditionally detectable by $(x, (Px)^3)$, the fault masking probability is only a bit smaller than the best fault masking probability of multilinear codes. Given the fact that $(x, (Px)^3)$ code requires much more hardware overhead to implement, we claim multilinear codes are more promising alternatives in practice.

4.2 Multilinear Arithmetic Codes

In this section, two constructions of multilinear arithmetic codes will be presented. Different from the widely used non-systematic AN codes (Rao and Garcia, 1971), the

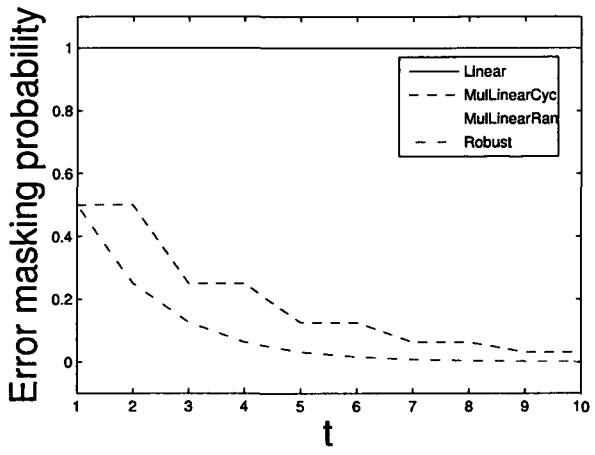


Figure 4·2: Comparison of fault masking probability after t clock cycles

codewords of systematic arithmetic codes contain two parts: the information part and the redundant part. Any codeword c can be written in the format of $(x, y), x \in Z_{2^k}, y \in Z_{2^r}$, where k is the number of information bits, r is the number of redundant bits and Z_{2^k} is the additive group of integers $\{0, 1, \cdots, 2^k - 1\}$. Faults in arithmetic devices usually manifest as arithmetic errors at the output of the device. We denote by $e = (e_x, e_y)$ the error vector and $\tilde{c} = (|x + e_x|_{2^k}, |y + e_y|_{2^r})$ the distorted codeword in which $e_x \in Z_{2^k}, e_y \in Z_{2^r}$, $+$ is the arithmetic addition and $|\cdot|_p$ is the modulo p operation.

Let $C = \{(x, y)\}, x \in GF(2^k), y \in GF(2^r)$ be an arithmetic code. An error $e = (e_x, e_y)$ is masked by a codeword $c = (x, y) \in C$ if $\tilde{c} = (|x + e_x|_{2^k}, |y + e_y|_{2^r})$ also belongs to C. Given an error e, the error masking probability $Q(e)$ is calculated as follows:

$$Q(e) = \frac{\{c|c \in C, \tilde{c} \in C\}}{|C|}, \tag{4.1}$$

where $|C|$ is the size of C. If an error is masked by all codewords of the code, $Q(e) = 1$ and the error is called **undetectable**. If $0 < Q(e) < 1$, the error is called **conditionally detectable**. Different from algebraic codes, arithmetic codes

rarely have undetectable errors. To illustrate the advantage of multilinear arithmetic codes, we compare the number and the probability of **bad errors** – errors e with $Q(e) \geq 0.5$ – for linear arithmetic codes and the proposed multilinear arithmetic codes. (This definition of bad errors was also used in (Kulikowski et al., 2008b).) Since bad errors are the most difficult to detect, we will show that the transition from linear to multilinear arithmetic codes results in a drastic reduction of the number of bad errors and an improvement of the error detection ability of the code (see Section 4.2.1).

Remark 4.2.1 *Estimations of numbers of bad errors presented in this paper can be easily generalized to the case when bad errors e are defined as errors with $Q(e) \geq \beta$ for any $\beta > 0$.*

4.2.1 Linear and Partially Robust Arithmetic Codes

We first analyze the error detection properties of linear arithmetic codes.

Theorem 4.2.1 ***(Linear Arithmetic Codes)*** *Let C be a linear arithmetic code defined by*

$$C = \{(x, y) | x \in Z_{2^k}, y = f(x) \in Z_{2^r}\}, \tag{4.2}$$

in which $f(x) = |x|_p$, p is an integer (not a power of 2) and $|\cdot|_p$ represents the modulo p reduction operation. Denote by $e = (e_x, e_y)$ an additive error, $e_x \in Z_{2^k}, e_y \in Z_{2^r}$, $r = \lceil \log_2 p \rceil$. The distorted codeword is $\tilde{c} = c + e = (|x + e_x|_{2^k}, |y + e_y|_{2^r})$, $c \in C$. As $\frac{p}{2^k} \to 0$, the number of bad errors converges to

$$2 \cdot (2^k - 2^k(H_{p-1} - H_{\lfloor \frac{p}{2} \rfloor}) - \frac{2^{k-1}}{p}), \tag{4.3}$$

where $H_n = \sum_{i=1}^{n} \frac{1}{i}$ represents the n-th harmonic number. For large p the difference $H_{p-1} - H_{\lfloor \frac{p}{2} \rfloor}$ converges to $\ln 2$. In this case the probability of bad errors converges to $0.3 \cdot 2^{-r+1}$ as $\frac{p}{2^k} \to 0$.

If no errors occur to the redundant part of the code ($e_y = 0$), the number of bad errors $e = (e_x, 0)$ is upper bounded by $2 \cdot \lceil \frac{2^{k-1}}{p} \rceil$ and is lower bounded by $2 \cdot \lfloor \frac{2^{k-1}}{p} \rfloor$. As $\frac{p}{2^k} \to 0$, the probability of bad errors in the format of $e = (e_x, 0)$ converges to p^{-1}.

Proof To simplify the analysis, we divide the errors into two classes according to the value of $x + e_x$.

1. $x + e_x < 2^k$, we have $|x + e_x|_{2^k} = x + e_x$, $f(x + e_x) = |x + e_x|_p$.

   (a) $|x|_p + e_y < p$, then $||x|_p + e_y|_{2^r} = |x|_p + e_y$. An error (e_x, e_y) is masked if and only if $|x+e_x|_p = |x|_p+e_y$. Or equivalently $|e_x|_p = e_y$. For a codeword x to mask a given error (e_x, e_y), the following conditions must be satisfied:

$$x + e_x < 2^k, \tag{4.4}$$

$$|x|_p + e_y < p, \tag{4.5}$$

$$|e_x|_p = e_y. \tag{4.6}$$

   From (4.5) and (4.6) we have $|x|_p < p-|e_x|_p$. For any given $|x|_p < p-|e_x|_p$, the number of x satisfying (4.4) is upper bounded by $\lceil \frac{2^k-e_x}{p} \rceil$. Thereby for a given error (e_x, e_y), the total number of codewords that mask the error is upper bounded by $\lceil \frac{2^k-e_x}{p} \rceil \cdot (p - |e_x|_p)$. For bad errors the error masking probability is larger or equal to 0.5. Thus

$$2^{-k} \cdot \lceil \frac{2^k - e_x}{p} \rceil \cdot (p - |e_x|_p) \geq 0.5. \tag{4.7}$$

   As $\frac{p}{2^k} \to 0$, the asymptotic error masking probability can be estimated by removing the ceiling function. Thereby (4.7) can be re-written as follows:

$$2^{-k} \cdot \frac{2^k - e_x}{p} \cdot (p - |e_x|_p) \geq 0.5. \tag{4.8}$$

   Thereby,

$$e_x \leq \frac{2^{k-1}(p - 2 \cdot |e_x|_p)}{p - |e_x|_p}. \tag{4.9}$$

We know that $e_x \geq 0$, so

$$0 \leq |e_x|_p \leq \lfloor \frac{p}{2} \rfloor. \tag{4.10}$$

The total number of e_x satisfying (4.9) and (4.10) is upper bounded by (For simplicity, let $i = |e_x|_p$.)

$$\sum_{i=0}^{\lfloor \frac{p}{2} \rfloor} (\frac{1}{p} \cdot \frac{2^{k-1}(p-2i)}{p-i} + 1), \tag{4.11}$$

and is lower bounded by

$$\sum_{i=0}^{\lfloor \frac{p}{2} \rfloor} (\frac{1}{p} \cdot \frac{2^{k-1}(p-2i)}{p-i} - 1), \tag{4.12}$$

So the number of bad errors in this class is upper bounded by (4.11) and is lower bounded by (4.12).

(b) $p \leq |x|_p + e_y < 2^r$, errors in this class will never be masked because the redundant part of a distorted codeword is an invalid value.

(c) $|x|_p + e_y \geq 2^r$, then $||x|_p + e_y|_{2^r} = |x|_p + e_y - 2^r$. An error (e_x, e_y) is masked if and only if $|x + e_x|_p = |x|_p + e_y - 2^r$. Or equivalently $|e_x|_p = |e_y - 2^r|_p$. It is easy to show that $e_y - 2^r \in [-p+1, -1]$, so $|e_y - 2^r|_p = p + e_y - 2^r$. For a codeword x to mask a given error (e_x, e_y), the following conditions must be satisfied:

$$x + e_x < 2^k, \tag{4.13}$$

$$|x|_p + e_y \geq 2^r, \tag{4.14}$$

$$|e_x|_p = e_y + p - 2^r. \tag{4.15}$$

From (4.14) and (4.15) we have $|x|_p \geq p - |e_x|_p$. For a certain value of

$|x|_p \geq p - |e_x|_p$, the number of x satisfying (4.13) is upper bounded by $\lceil \frac{2^k - e_x}{p} \rceil$. Thereby for a given error (e_x, e_y), the total number of codewords that mask the error is upper bounded by $\lceil \frac{2^k - e_x}{p} \rceil \cdot |e_x|_p$. For bad errors the error masking probability is larger or equal to 0.5. Thus

$$2^{-k} \cdot \lceil \frac{2^k - e_x}{p} \rceil \cdot |e_x|_p \geq 0.5. \tag{4.16}$$

As $\frac{p}{2^k} \to 0$, the error masking probability can be estimated by removing the ceiling function. Thereby (4.16) can be re-written as follows:

$$2^{-k} \cdot \frac{2^k - e_x}{p} \cdot |e_x|_p \geq 0.5. \tag{4.17}$$

So

$$e_x \leq \frac{1}{|e_x|_p} \cdot 2^{k-1} \cdot (2|e_x|_p - p). \tag{4.18}$$

Because $e_x \geq 0$,

$$|e_x|_p \geq \lceil \frac{p}{2} \rceil. \tag{4.19}$$

The total number of e_x satisfying (4.18) and (4.19) is upper bounded by (For simplicity, let $i = |e_x|_p$.)

$$\sum_{\lceil \frac{p}{2} \rceil}^{p-1} (\frac{1}{p} \cdot \frac{2^{k-1}(2i - p)}{i} + 1), \tag{4.20}$$

and is lower bounded by

$$\sum_{\lceil \frac{p}{2} \rceil}^{p-1} (\frac{1}{p} \cdot \frac{2^{k-1}(2i - p)}{i} - 1). \tag{4.21}$$

So the number of bad errors in this class is upper bounded by (4.20) and is lower bounded by (4.21).

From the above analysis, the total number of bad errors for the case when $x + e_x < 2^k$ is upper bounded by $2^k + p - 2^k \sum_{i=\lceil \frac{p}{2} \rceil}^{p-1} \frac{1}{i} - \frac{2^{k-1}}{p}$ and is lower bounded by $2^k - p - 2^k \sum_{i=\lceil \frac{p}{2} \rceil}^{p-1} \frac{1}{i} - \frac{2^{k-1}}{p}$.

2. $x + e_x \geq 2^k$, we have $|x + e_x|_{2^k} = x + e_x - 2^k$, $f(x + e_x) = |x + e_x - 2^k|_p$. Following the same analysis, we can show that the number of bad errors in this class is also upper bounded by $2^k + p - 2^k \sum_{i=\lceil \frac{p}{2} \rceil}^{p-1} \frac{1}{i} - \frac{2^{k-1}}{p}$ and lower bounded by $2^k - p - 2^k \sum_{i=\lceil \frac{p}{2} \rceil}^{p-1} \frac{1}{i} - \frac{2^{k-1}}{p}$.

Thereby for linear arithmetic codes, an upperbound of the number of bad errors is

$$\begin{aligned} & 2 \cdot (2^k + p - 2^k \sum_{i=\lceil \frac{p}{2} \rceil}^{p-1} \frac{1}{i} - \frac{2^{k-1}}{p}) \\ = \; & 2 \cdot (2^k + p - 2^k (H_{p-1} - H_{\lfloor \frac{p}{2} \rfloor}) - \frac{2^{k-1}}{p}). \end{aligned}$$

Similarly, a lowerbound of the number of bad errors is

$$2 \cdot (2^k - p - 2^k (H_{p-1} - H_{\lfloor \frac{p}{2} \rfloor}) - \frac{2^{k-1}}{p}).$$

As $\frac{p}{2^k} \to 0$, the number of bad errors converges to (4.3).

If no errors occur to the redundant part of the code, $e_y = 0$. For the case when $x + e_x < 2^k$, a codeword x mask an error $e = (e_x, e_y = 0)$ if and only if $|e_x|_p = e_y = 0$. It is easy to prove that the number of errors in this class is upper bounded by $\lceil \frac{2^{k-1}}{p} \rceil$ and is lower bounded by $\lfloor \frac{2^{k-1}}{p} \rfloor$. Similarly, when $x + e_x \geq 2^k$, the number of bad errors in the format of $(e_x, 0)$ is also upper bounded by $\lceil \frac{2^{k-1}}{p} \rceil$ and is lower bounded by $\lfloor \frac{2^{k-1}}{p} \rfloor$. So the total number of bad errors occurring to the information part of the code is between $2 \cdot \lfloor \frac{2^{k-1}}{p} \rfloor$ and $2 \cdot \lceil \frac{2^{k-1}}{p} \rceil$. ■

For linear arithmetic codes, the number of bad errors in the format of $e = (e_x, 0)$ decreases as p increases. When $p > 2^{k-1}$, there are very few bad errors in the format

of $e = (e_x, 0)$. However, the total number of bad errors is still very large for linear arithmetic codes.

In general, the hardware overhead for the encoder of the code is mostly affected by the number of redundant bits $r = \lceil \log_2 p \rceil$. The smallest fraction of bad errors for linear arithmetic code is of the order of 2^{-r}. The only way to reduce the fraction is to increase the number of redundant bits, which is costly in terms of the hardware overhead. To reduce the number of bad errors while maintaining the number of redundant bits, partially robust codes based on nonlinear functions were proposed in (Gaubatz et al., 2006; Kulikowski et al., 2008b).

Construction 4.2.1 *(Gaubatz et al., 2006; Kulikowski et al., 2008b) Let $x \in GF(2^k)$, p be a prime number larger than 2 and $r = \lceil log_2 p \rceil$. Denote by $|\cdot|_p$ the modulo p reduction operation. The arithmetic code C composed of all vectors $(x, |x^2|_p)$, in which $|x^2|_p \in GF(2^r)$, is a partially robust arithmetic code.*

Partially robust $(x, |x^2|_p)$ codes have nearly no bad errors and can provide better protection of cryptographic devices than linear arithmetic codes assuming a slow fault-injection mechanism (Kulikowski et al., 2008b). However, $(x, |x^2|_p)$ codes rely on nonlinear squaring operations and have larger overhead than linear arithmetic codes. Moreover, $(x, |x^2|_p)$ codes have worse detection capabilities of errors in the format of $e = (e_x, 0)$ (Chapter 5).

We next propose two constructions of multilinear arithmetic codes based on the idea of randomly selecting among multiple linear arithmetic codes for each encoding and the corresponding decoding operation. For each multiplication, one randomly selected code is used to generate the redundant bits and decode the possibly distorted outputs of the multiplier and the predictor. For different multiplications, different codes may be used.

Intuitively, when we randomly select among multiple codes, even if an error is missed by one of the codes, it may still be detected by other codes. Suppose we

randomly select among L codes with equal probabilities. Let $p_i(e), 1 \leq i \leq L$ be the probability that an error e is masked by the i^{th} code. It is easy to show that the average probability that the error e is masked when we randomly select one out of these L codes with the same probability can be computed as

$$p(e) = \sum_{i=1}^{L} p_i(e)/L \tag{4.22}$$

With different error detecting properties, the L codes will have different distribution of error masking probabilities $p_i(e), 1 \leq i \leq L$. When randomly selecting among them, even if some $p_i(e)$ are larger than 0.5 (For single arithmetic codes, the error is bad.), it is highly probable that the average error masking probability $p(e)$ will still be smaller than 0.5 due to the fact that other other codes can detect the error with a high probability. Specifically, when we randomly select from two codes, the only possible bad errors are errors masked by both of the codes or errors masked by one code with probability one. Obviously, this constrain will drastically reduce the number of bad errors.

The multilinear codes proposed in the left part of this section have similar number of bad errors to $(x, |x^2|_p)$ partially robust codes. One construction will result in a hardware overhead close to architectures based on linear arithmetic codes. The other construction will have much better error detection capabilities of errors in the format of $e = (e_x, 0)$ than linear and partially robust arithmetic codes.

4.2.2 $[|x|_p, |2x|_p]$ Multilinear Code

Theorem 4.2.2 *Let C_1, C_2 be two arithmetic systematic codes defined by*

$$C_i = \{(x, y) | x \in Z_{2^k}, y = f_i(x) \in Z_{2^r}\}, i \in \{1, 2\},$$

where $f_1(x) = |x|_p$, $f_2(x) = |2x|_p$. Denote by $e = (e_x, e_y)$ the arithmetic errors and $\tilde{c} = c + e = (|x + e_x|_{2^k}, |y + e_y|_{2^r})$ the distorted codeword, where $e_x \in Z_{2^k}, e_y \in Z_{2^r}$

and $r = \lceil \log_2 p \rceil$ *is the number of redundant bits. If we randomly select* C_1 *and* C_2 *to encode the original messages with equal probability, the total number of bad errors is upper bounded by* $2 \cdot \lceil \frac{2^{k-1}}{p} \rceil$ *and is lower bounded by* $2 \cdot \lfloor \frac{2^{k-1}}{p} \rfloor$. *As* $\frac{p}{2^k} \to 0$, *the probability of bad errors for* $[|x|_p, |2x|_p]$ *multilinear codes converges to* $2^{-r}p^{-1}$. *The probability of bad errors in the format of* $(e_x, 0)$ *converges to* p^{-1}.

Proof A non-zero error e is masked by a linear arithmetic code when one of the four cases shown in Table 4.3 is satisfied. (Please refer to the proof of Theorem 4.2.1 for more details.) When we randomly select C_1 and C_2 with equal probability, an error

Table 4.3: Classification of masked errors for linear arithmetic codes

Case1	Case2	Case3	Case4
$x + e_x < 2^k$ $f_i(x) + e_y < p$ $f_i(e_x) = e_y$	$x + e_x < 2^k$ $f_i(x) + e_y \geq 2^r$ $f_i(e_x) = e_y + p - 2^r$	$x + e_x \geq 2^k$ $f_i(x) + e_y < p$ $f_i(e_x - 2^k) = e_y$	$x + e_x \geq 2^k$ $f_i(x) + e_y \geq 2^r$ $f_i(e_x - 2^k) = e_y + p - 2^r$

$e = (e_x, e_y)$ is bad if only it is masked by both of the codes or it is masked by one code with probability 1. More specifically, e is bad if and only if the total number of codewords in C_1 and C_2 that mask e is larger or equal to 2^k.

1. For a given error e, when C_1 is in Case1 (i.e. $x+e_x < 2^k, f_1(x)+e_y < p, f_1(e_x) = e_y$) or Case2 and C_2 is in Case3 or Case4, the total number of codewords masking the error e is less than 2^k. So there are no bad errors in this class. Similarly, when C_1 is in Case3 or Case4 and C_2 is in Case1 or Case2, there are no bad errors.

2. C_1 is in Case1 and C_2 is in Case2. For C_1, $|x|_p + e_y < p$, the possible number of $|x|_p$ is $p - e_y$. For C_2, $|2x|_p + e_y \geq 2^r$, the possible number of $|x|_p$ is $p - 2^r + e_y$. For each possible value of $|x|_p$, the number of x masking the error is upper bounded by $\lceil \frac{2^k - e_x}{p} \rceil$. It is easy to prove that the total number of x masking the error is less than 2^k. So there are no bad errors in this class. Similarly we can prove that for the following three cases there are also no bad errors.

(a) C_1 is in Case2, C_2 is in Case1;

(b) C_1 is in Case3, C_2 is in Case4;

(c) C_1 is in Case4, C_2 is in Case3.

3. When C_1 and C_2 both belong to Case2, $x+e_x < 2^k$, we have $|x+e_x|_{2^k} = x+e_x$, $|x|_p + e_y \geq 2^r$ and $|2x|_p + e_y \geq 2^r$. In this case $||x|_p + e_y|_{2^r} = |x|_p + e_y - 2^r$, $||2x|_p + e_y|_{2^r} = |2x|_p + e_y - 2^r$. For C_1, an error (e_x, e_y) is missed if and only if

$$|x + e_x|_p = |x|_p + e_y - 2^r \Rightarrow |e_x|_p = |e_y - 2^r|_p. \tag{4.23}$$

For C_2, an error (e_x, e_y) is missed if and only if

$$|2 \cdot (x + e_x)|_p = |2x|_p + e_y - 2^r \Rightarrow |2e_x|_p = |e_y - 2^r|_p. \tag{4.24}$$

From (4.23) and (4.24) we have $|e_x|_p = |e_y - 2^r|_p = e_y + p - 2^r = 0$. For an error to be masked by both of the codes, the following conditions must be satisfied:

$$x + e_x < 2^k, \tag{4.25}$$

$$|x|_p + e_y \geq 2^r, \tag{4.26}$$

$$|2x|_p + e_y \geq 2^r, \tag{4.27}$$

$$|e_x|_p = e_y + p - 2^r = 0. \tag{4.28}$$

From (4.28), $e_y = 2^r - p \Rightarrow |x|_p + e_y < 2^r, |2x|_p + e_y < 2^r$. So no errors in this case will be masked by both of the codes. Errors in this class are all non-bad errors. Similarly, when C_1 and C_2 both belong to Case4, there are no bad errors.

4. When C_1 and C_2 both belong to Case1, $x+e_x < 2^k$, we have $|x+e_x|_{2^k} = x+e_x$, $f_1(x+e_x) = |x+e_x|_p$, $f_2(x+e_x) = |2\cdot(x+e_x)|_p$. $|x|_p+e_y < p$ and $|2x|_p+e_y < p$. In this case $||x|_p + e_y|_{2^r} = |x|_p + e_y$, $||2x|_p + e_y|_{2^r} = |2x|_p + e_y$. For C_1, an error

(e_x, e_y) is missed if and only if

$$|x + e_x|_p = |x|_p + e_y \Rightarrow |e_x|_p = e_y. \tag{4.29}$$

For C_2, an error (e_x, e_y) is missed if and only if

$$|2 \cdot (x + e_x)|_p = |2x|_p + e_y \Rightarrow |2e_x|_p = e_y. \tag{4.30}$$

From (4.29) and (4.30) we have $|e_x|_p = e_y = 0$. For a codeword x to mask a given error (e_x, e_y), the following conditions must be satisfied:

$$x + e_x < 2^k, \tag{4.31}$$

$$|x|_p + e_y < p, \tag{4.32}$$

$$|2x|_p + e_y < p, \tag{4.33}$$

$$|e_x|_p = e_y = 0. \tag{4.34}$$

When (4.34) is satisfied, (4.32) and (4.33) are also satisfied. For each (e_x, e_y) such that $|e_x|_p = e_y = 0$, the total number of codewords in C_1 and C_2 that mask the error is $2 \cdot (2^k - e_x)$. For bad errors this number should be larger or equal to 2^k. Thus

$$2 \cdot (2^k - e_x) \geq 2^k \tag{4.35}$$

$$\Rightarrow \quad e_x \leq 2^{k-1} \tag{4.36}$$

From (4.34) and (4.36), the number of non-zero bad errors is upper bounded by $\lceil \frac{2^{k-1}}{p} \rceil$ and is lower bounded by $\lfloor \frac{2^{k-1}}{p} \rfloor$. Similarly, when C_1 and C_2 both belong to Case3, the number of bad errors is upper bounded by $\lceil \frac{2^{k-1}}{p} \rceil$ and is lower bounded by $\lfloor \frac{2^{k-1}}{p} \rfloor$.

So the total number of bad errors is between $2 \cdot \lceil \frac{2^{k-1}}{p} \rceil$ and $2 \cdot \lfloor \frac{2^{k-1}}{p} \rfloor$. ■

The number of bad errors for $[|x|_p, |2x|_p]$ multilinear codes is much smaller than that for the linear arithmetic codes (see (4.3)). All bad errors are in the format of $e = (e_x, 0)$. The security level of systems protected by the $[|x|_p, |2x|_p]$ codes can be further increased by implementing a merged design of the original device and the encoder generating redundant bits of the output of the protected device. In that case, the injected faults will have high probability to affect not only the original device but also the encoder that generates the redundant bits of the code. The probability of errors in the format of $e = (e_x, 0)$ will be efficiently reduced. As a result, the error detection capabilities and the security level of the system will be increased.

If the original device and the encoder are separated and the attacker is able to inject faults only in the original device, $[|x|_p, |2x|_p]$ multilinear codes do not have any advantages over linear arithmetic codes in terms of the error detecting capability. In this case, the system should be protected using multi-modulii multilinear codes shown below.

A more general case of Theorem 4.2.2 is to randomly select from $L \leq p-1$ codes defined by $C_i = \{x, f_i(x)\}$ where $f_i(x) = |ix|_p, 1 \leq i < p$. However, it is easy to show that increasing the number of codes from which we randomly select a code for encoding and decoding will not reduce the number of bad errors in this situation.

We next present a construction based on using multiple modulii. The resulting codes will be different from $[|x|_p, |2x|_p]$ codes in the following two aspects.

- They have much less bad errors in the format of $e = (e_x, 0)$;
- Increasing the number of codes from which we randomly select a code for encoding and decoding will further reduce the number of bad errors.

4.2.3 Multi-modulii Multilinear Code

Theorem 4.2.3 *Let C_1, C_2 be two systematic arithmetic codes defined by*

$$C_i = \{(x, y) | x \in Z_{2^k}, y = f_i(x) \in Z_{2^r}\}, i \in \{1, 2\},$$

in which $f_1(x) = |x|_p$, $f_2(x) = |x|_q$ where p, q are co-prime numbers (not a power of 2) and $r = \max(\lceil \log_2 p \rceil, \lceil \log_2 q \rceil)$. If we randomly select C_i with equal probability to encode the original messages, the number of bad errors in the format of $e = (e_x, 0)$ is upper bounded by

$$2(\lceil \frac{2^{k-1}}{pq} \rceil + \lceil \frac{2^k}{pq} \rceil), \tag{4.37}$$

and is lower bounded by

$$2(\lfloor \frac{2^{k-1}}{pq} \rfloor + \lfloor \frac{2^k}{pq} \rfloor). \tag{4.38}$$

When $pq << 2^k$, the probability of bad errors in the format of $(e_x, 0)$ converges to $3p^{-1}q^{-1}$.

Proof Since $e_y = 0$, we have $|x|_p + e_y < p$ and $|x|_q + e_y < q$. For each linear code, errors are masked if and only if one of the following two conditions are satisfied (see the proof of Theorem 4.2.1).

Case1 : $x + e_x < 2^k, f_i(e_x) = 0$.

Case2 : $x + e_x \geq 2^k, f_i(e_x - 2^k) = 0$.

If we randomly select C_i with equal probability, an error $e = (e_x, 0)$ is masked by a probability at least 0.5 if and only if the total number of codewords belonging to C_i which mask the error is larger or equal to 2^k. For a given non-zero error $e = (e_x, 0)$, there are three possible situations as stated below.

1. Both C_i belong to Case1, $|e_x|_p = |e_x|_q = 0$. The total number of codewords belonging to C_i which mask the error e is $2 \cdot (2^k - e_x)$. Hence the error is bad if and only if $e_x \leq 2^{k-1}$. Since $|e_x|_p = |e_x|_q = 0$, the number of bad errors in this

class is upper bounded by $\lceil \frac{2^{k-1}}{pq} \rceil$ and is lower bounded by $\lfloor \frac{2^{k-1}}{pq} \rfloor$ ($e_x = 2^{k-1}$ does not satisfy $|e_x|_p = 0$ and $|e_x|_q = 0$).

2. Both C_i belong to Case2, following similar analysis we can prove that the number of bad errors in this class is upper bounded by $\lceil \frac{2^{k-1}}{pq} \rceil$ and is lower bounded by $\lfloor \frac{2^{k-1}}{pq} \rfloor$.

3. When C_i belong to different cases, it is easy to prove that as long as $e = (e_x, 0)$ satisfies $|e_x|_p = 0, |e_x|_q = |2^k|_q$ or $|e_x|_p = |2^k|_p, |e_x|_q = 0$, the total number of codewords masking the error is always 2^k. Hence the error is always bad. The number of bad errors in this class is upper bounded by $2\lceil \frac{2^k}{pq} \rceil$ and is lower bounded by $2\lfloor \frac{2^k}{pq} \rfloor$.

Remark 4.2.2 *When randomly selecting from two codes, experimental results show that the total number of bad errors $e = (e_x, e_y)$ for the multi-modulii codes is comparable to that of $[|x|_p, |2x|_p]$ multilinear codes and is much smaller than that of linear arithmetic codes. The idea of using multiple residues as the redundant part of the code has already been presented in (Rao and Garcia, 1971). With two residues, the codeword was in the format of $(x, |x|_p, |x|_q)$. We want to emphasize that our construction is different from multi-residue codes proposed in (Rao and Garcia, 1971) since at each clock cycle our code has only one residue for the redundant part. Instead of using multiple residues simultaneously, we use only one for each encoding and decoding operation and randomly select the modulus for different operations.*

Table 4.4: Number of bad errors in the format of $e = (e_x, 0)$ for linear and multilinear codes ($k = 32$)

	$p = 5$	$p = 241$	$p = 563$	$p = 883$	$p = 1237$	$p = 2767$
LinearArithmetic	8.6×10^8	1.8×10^7	7.6×10^6	4.9×10^6	3.5×10^6	1.6×10^6
$[\|x\|_p, \|2x\|_p]$ multilinear codes	8.6×10^8	1.8×10^7	7.6×10^6	4.9×10^6	3.5×10^6	1.6×10^6
Multi-modulii codes ($L = 2$)	8.6×10^8	2.2×10^5	4.1×10^4	1.7×10^4	8.5×10^3	1.7×10^3

Multi-modulii codes have much less bad errors in the format of $e = (e_x, 0)$ than linear and $[|x|_p, |2x|_p]$ multilinear arithmetic codes (see (4.37)). Table 4.4 shows the

estimated number of bad errors in this class for all three constructions. The number of information bits of the codes in the Table is $k = 32$. For multi-modulii codes, p corresponds to the larger modulii. The other modulii is selected to be the largest possible prime number less than p, e.g. when $p = 241$, the other modulii is 239. As p increases, the number of bad errors for multi-modulii codes decreases much faster than for the other two alternatives. When $p = 2767(r = 12)$, the multi-modulii code has only 1.7×10^3 bad errors in the format of $(e_x, 0)$ while the other two codes have about 1.6×10^6.

These characteristics of multi-modulii codes are beneficial in many different situations. For example, when the attacker can identify and inject faults only to the original device, or the encoder of the code is only a small part of the cryptographic system and is separated from the original device so that most of the injected single (or even double) faults affect only the original device, errors will be in the format of $e = (e_x, 0)$. In this case, systems protected by multi-modulii codes will have a higher security level than architectures based on other alternatives. Systems with different error rates for the original device and the predictor generating the redundant bits $y = f(x)$ can also benefit from this characteristic of multi-modulii codes. Design based on multi-voltage regions is proposed to reduce the total power consumption of the system (Khursheed et al., 2009). In the region with the smaller voltage level, circuits are more vulnerable to soft errors and are more probable to have errors caused by problems such as timing violations (Roberts et al., 2005). As a result, the error rate for circuits in this region will be higher. If the original device operates at a lower voltage level than the predictor, multi-modulii codes can provide better protections due to the fact that they have higher detection capabilities of errors in the format of $e = (e_x, 0)$.

Different from $[|x|_p, |2x|_p]$ codes, for multi-modulii codes increasing the number of codes from which we randomly select a code for encoding and decoding can further reduce the total number of bad errors. Table 4.5 shows the simulation results for a 8-bit multipliers protected by multi-modulii codes with different number of modulii. The second line corresponds to the case when a single linear arithmetic code is used. When we randomly select from multiple linear arithmetic codes with four different modulii, the number of bad errors in the format of $e = (e_x, 0)$ is only 13, which is more than 100 times better than architectures based on linear arithmetic codes.

Table 4.5: Number of bad errors when selecting from linear arithmetic codes with different modulii

Modulii	Bad Errors	Bad Errors $e = (e_x, 0)$
$p_1 = 31$	47857	2113
$p_1 = 31, p_2 = 29$	1781	249
$p_1 = 31, p_2 = 29, p_3 = 23$	1651	180
$p_1 = 31, p_2 = 29, p_3 = 23, p_4 = 19$	133	13

Remark 4.2.3 *From Table 4.5, when we randomly select from two linear arithmetic codes with different modulii, the number of bad errors in the format of* $e = (e_x, 0)$ *is a little bit larger than the result given by (4.37). This is because when using arithmetic codes to protect multipliers, the output of the multiplier, hence the information bits of the arithmetic codes, is not uniformly distributed. Moreover, some combinations of information bits in* Z_{2^k} *may never occur at the output of a multiplier whose operands are* $\frac{k}{2}$ *bits. However, simulation results show that in this situation multilinear codes still largely over-perform linear arithmetic codes and all the advantages of multilinear codes are preserved.*

Table 4.6 summarizes the probability of bad errors of linear and multilinear arithmetic codes. The hardware design of secure multiplier architectures based on $[|x|_p, |2x|_p]$ and multi-modulii codes as well as linear and $(x, |x^2|_p)$ codes will be presented in Chapter 5.

Table 4.6: Probability of bad errors for linear and multilinear codes

Probability of	Linear Arithmetic Codes	$[\lvert x\rvert_p, \lvert 2x\rvert_p]$ codes	Multi-modulii codes
Bad errors	$\approx 0.3 \cdot 2^{-r+1}$	$\approx 2^{-r}p^{-1}$	$\approx 2^{-r}p^{-1^*}$ $(q \approx p)$
Bad errors $e = (e_x, 0)$	$\approx p^{-1}$	$\approx p^{-1}$	$\approx 3p^{-1}q^{-1}$

* : Based on experimental results.

4.3 Summary

In this Chapter, we present constructions of both algebraic and arithmetic multilinear codes. The basic idea of multilinear codes is to randomly select from a given set of linear codes for each encoding and the corresponding decoding operation. Even if an error e cannot be detected by one of the linear codes, the chance that the multilinear code will miss this error can still be very small due to the fact that the error can be detected by other codes with high probabilities.

Compared to robust and partially robust codes, multilinear codes achieve similar error detection capabilities at much reduced cost due to their linearity. The performance of multilinear codes does not depend on the distribution of the spatial multiplicities of errors and improves as the errors last for more than one clock cycle.

Chapter 5

Application of Robust Codes on the Design of Secure Cryptographic Devices

Cryptographic devices are widely used in applications like ATM cards and commercial electronics. These devices are vulnerable to side-channel attacks such as timing-analysis attacks (Kocher, 1996), power-analysis attacks (Kocher et al., 1999) and fault-injection attacks (Skorobogatov and Anderson, 2003; Bar-El et al., 2006). Due to their active and adaptive nature, fault-based attacks are one of the most powerful types of side-channel attacks. Since a fault attack was demonstrated by Boneh et al. in (Boneh et al., 2001), numerous papers have been published proposing a variety of fault attacks on both public-key and private-key cryptographic devices. One of the most efficient fault-injection attacks on AES-128, for example, requires only two faulty ciphertexts to retrieve all 128 bits of the secret key (Piret and Quisquater, 2003). Without proper protection against fault-injection attacks, the security of cryptographic devices can never be guaranteed.

In (Maistri and Leveugle, 1982; Maistri et al., 2007), a solution based on time redundancy by means of a double-data-rate (DDR) computation template was presented. Each computation is conducted twice and the results are compared to detect injected faults. Both clock edges were exploited to control the computation flow for the purpose of improving the throughput of the system. In (Moore et al., 2002; Kulikowski et al., 2008e), the authors investigated the usage of dual-rail encoding for the

protection of cryptographic devices against different types of side-channel attacks in asynchronous circuits.

The most commonly used fault detection technique is concurrent error detection (CED) which employs circuit level coding techniques, e.g. parity schemes, modular redundancy, etc. to produce and verify check digits after each computation. In (Bertoni et al., 2003), a secure AES architecture based on linear parity codes was proposed. The method can detect all errors of odd multiplicities with reasonable hardware overhead. In (Gaubatz and Sunar, 2006), an approach to fault tolerant public key cryptography based on redundant arithmetic in finite rings were presented. The method is closely related to cyclic binary and arithmetic codes. In (Karri et al., 2002), the authors proposed a CED technique that exploits the inverse relationships existing between encryption and decryption at various levels. A decryption is immediately conducted to verify the correctness of the encryption operation. A lightweight concurrent fault detection scheme for the S-box of AES was proposed in (Kermani and Reyhani-Masoleh, 2008). The structure of the S-box is divided into blocks and the predicted parities for these blocks are obtained and used for the fault detection. Various fault attack countermeasures were compared in terms of the hardware overhead and the the fault detection capabilities in (Malkin et al., 2006).

Most of the proposed error detecting codes are linear codes like parity codes, Hamming codes and AN codes. Protection architectures based on linear codes concentrate their error detecting abilities on errors with small multiplicities or errors of particular types, e.g. errors with odd multiplicities or byte errors. However, in the presence of unanticipated types of errors linear codes can provide little protection. Linear parity codes, for example, can detect no errors with even multiplicities.

In (Bousselam et al., 2010), the author compared several concurrent fault detection schemes for advanced encryption standard based on linear codes. The simulation

results showed the error detecting capabilities of systems protected by linear codes largely depend on the error profiles at the output of the device due to the injected faults. The spectrum of available fault injection methods and the adaptive nature of an attacker suggests that it would be possible to bypass such protection by injecting a class of faults or errors which the cryptographic device does not anticipate. Considering even only inexpensive non-invasive or semi-invasive fault attacks, there is a wide spectrum of the types of faults and injection methods an attacker has at his disposal (Bar-El et al., 2006).

In this Chapter, we propose robust codes and multilinear codes as solutions to the limitation of linear error detecting codes for the protection of cryptographic devices against malicious fault injection attacks. Instead of concentrating the error detecting abilities on particular types of errors, these codes provide nearly equal protection against all error patterns, thus eliminate the weakness of linear codes which can be exploited by attackers to mount successful fault attacks. Moreover, the detection of errors for robust codes and multilinear codes are message-dependent. If the same error stays for more than one clock cycle, even if the injected fault manifests as an error that cannot be detected at the current clock cycle, it is still possible that the error will be detected at the next clock cycle when a new message arrives. Thereby, the advantage of robust codes and multilinear codes will be more significant for **lazy channels** where errors have high probabilities to repeat themselves for several clock cycles.

As case studies, we present the design of secure AES linear blocks using robust codes and partially robust codes, the construction of FSMs resilient to advanced fault injection attacks based on multilinear algebraic codes, and the design of robust multipliers (widely used as sub-blocks in public-key cryptosystems) using multilinear arithmetic codes. The error detection capability of architectures based on the pro-

posed codes was simulated and compared to architectures based on linear codes. All the designs were modeled in Verilog and synthesized in RTL design compiler. The overhead of different alternatives are estimated.

5.1 Fault and Attacker Model

Fault attacks can be performed in many different ways. The most investigated mechanisms of fault injections in the cryptography communities include introducing variations in power supplies (Canivet et al., 2010; Schmidt and Herbst, 2008; Kim and Quisquater, 2007; Barenghi et al., 2009), perturbing the silicon of the chip using white light or laser guns (light attacks) (Skorobogatov and Anderson, 2003; Monnet et al., 2006; Schmidt and Hutter, 2007; Canivet et al., 2010; Trichina and Korkikyan, 2010; Skorobogatov, 2010) and generating eddy current on the surface of the chip using magnetic field (electromagnetic attacks) (Samyde et al., 2002; Schmidt and Hutter, 2007), etc.

Fault attacks can be classified according to the capabilities of the attackers to control the parameters of the injected faults such as timing, locations, the type of the faults and the error patterns (Bar-El et al., 2006; Trichina and Korkikyan, 2010). With the vast arsenal of fault injection methods and techniques available to the attacker, the type of faults and the error patterns appearing as manifestations of the injected faults at the outputs of the device-under-attack is hard to model and predict. In (Barenghi et al., 2009), for example, the author showed that the number of faults can be controlled by reducing the supply voltage to a certain level. However, as the technology moves into deep-micro realm, it becomes harder and harder for the attacker to control the specific error patterns at the output of the device (Skorobogatov, 2010). Moreover, to the best of our knowledge, all the known fault injection mechanisms can only provide a limited spatial and timing resolution. For instance, the affected

die area due to a laser gun shot, which is one of the most powerful fault injection methods, is determined by the device technologies and the focus area of the laser beam (Skorobogatov, 2010). The time between two consecutive shot of the laser gun is affected by the speed of recharging and the delay between the trigger signal and the shot (Trichina and Korkikyan, 2010).

As in most papers on protecting the cryptographic devices, e.g. see (Kulikowski et al., 2008b; Malkin et al., 2006; Akdemir et al., 2011), we assume that countermeasures are implemented in the cryptographic device preventing the attackers from tampering with the clock signal (Bar-El et al., 2006) and the error detecting network (EDN). We further assume that a low-rate true random number generator (e.g. (Vasyltsov et al., 2008)) is available. In fact, most cryptographic devices incorporate a true random number generator by default for key initialization, random pad computation, challenge generation etc (Sunar et al., 2007b).

We assume a strong attacker model in which an attacker knows everything about the hardware architecture of the device including the codes used to detect errors. Specifically, the attackers may be able to inject faults which only affect the original device (but not the redundant portion used for error detection). We assume that the attacker cannot fully control the manifestation of injected faults as error patterns at the output of the device. However, as opposed to previous works on protecting cryptographic devices based on linear error detecting codes, e.g. parity codes or duplication codes, we do not impose any limitations on error patterns such as the number of distorted bits at the output of the protected device. The manifested error patterns are determined by factors such as the number of affected gates, input patterns to the device, etc. We further assume that the attacker cannot change the faults at each clock cycle (**slow fault-injection mechanisms**). Once faults are injected and an error is generated, the faults stay for several clock cycles before new faults can be

injected and tend to manifest themselves as the same error patterns at the output of the device. This is the case for several well known fault-injection methodologies mentioned in the last paragraph. We call this kind of channels where errors have high probabilities to repeat themselves for several consecutive clock cycles **lazy channels** or **channels with memory**. Multiple fault injections may also result in the same errors. For example, the author in (Trichina and Korkikyan, 2010) showed that two shots of a laser gun fired in rapid succession on a 32-bit ARM processor produces the same errors. The reason is that the fault locations cannot be adjusted in a short time due to the inflexible laser bench.

As it will be shown in the following sections, the advantages of robust codes and multilinear codes in terms of error detection capabilities are two-fold. First, they are better than linear codes in a sense that they have a much smaller number of undetectable errors (or bad errors for protecting arithmetic devices). Second, these codes have much higher error detection probabilities than linear codes for lazy channels hence they will effectively prevent the attacker from implementing a successful fault-induction attack under the aforementioned strong attacker model.

5.2 Repeatability of Errors

To support the statement that slow fault-injection methodologies may result in repeating errors, we conducted fault-injection simulations in C++ for unsigned and signed (2's complement) Wallace tree multipliers.

Multipliers based on Wallace trees (Wallace, 1964) are commonly used in various applications due to their faster speed compared to other alternatives. In general, the propagation delay of a n-bit Wallace tree multiplier is on the order of $O(log(n))$ in terms of logic gates. When combined with the Booth encoding technique, Wallace tree multipliers can be used for 2's complement multiplications. The gate level netlists for

both signed and unsigned Wallace tree multipliers are modeled in C++. In order to inject faults into the device, we insert a multiplexer at the output of every logic gate as shown in Figure 5·1. To simplify the analysis, we assume that the injected faults are either stuck-at-0 or stuck-at-1 faults. When *fault_enable* is asserted, *faulty_output* is selected and the observed output of the gate is determined by the internal fault model. We further assume that the attackers are able to inject multiple faults into the devices and the injected fault (or faults) may affect more than one logic gate. Ten thousands of simulations have been performed for every fixed number of affected gates. For each simulation, locations of the affected gates are randomly picked up and one million input operand pairs to the multipliers are randomly generated.

The injected faults may or may not manifest as non-zero arithmetic error patterns at the output of the multiplier. The error is in the format of $e = (e_x, e_y), e_x \in GF(2^k), e_y \in GF(2^r)$, where k is the number of information bits and r is the number of redundant bits. The probability of manifestation increases as more gates are affected. We also note that when only 1 gate is affected and the fault manifests, it will always manifest as the same non-zero error pattern at the output of the multiplier. Moreover, it is highly probable that the manifested non-zero error pattern is in the format of $\pm 2^i$, where i is an integer (single errors). As the number of the affected gates increases,

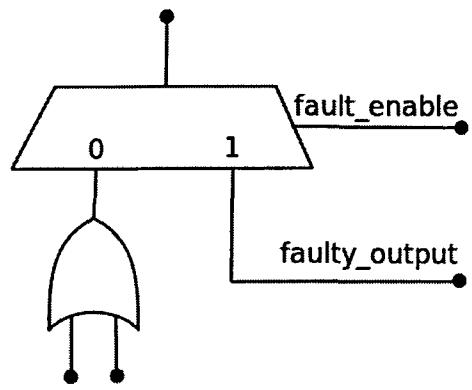


Figure 5·1: Fault-injection into a single gate

Table 5.1: The estimated repeatability of errors for faults injected into signed(2's complement) and unsigned Wallace tree multipliers

Type of the Multipliers	The number of faulty gates				
	1	2	3	4	5
16-bit Unsigned	0.5430	0.3166	0.2520	0.1857	0.1323
32-bit Unsigned	0.5174	0.2967	0.2301	0.1624	0.1110
16-bit Signed	0.3127	0.1904	0.1313	0.0839	0.0516
32-bit Signed	0.2730	0.1811	0.1197	0.0721	0.0418

both the number of possible error patterns and the average multiplicity of errors will increase.

For fixed faults, the error pattern $e = (e_x, e_y)$ observed at the output of the multiplier may vary for different input pairs. Assume that every multiplication takes one clock cycle to finish. Let e_t be the observed error pattern at the t^{th} clock cycle. The repeatability of errors can be defined by the following equation.

$$P_R = P(e_{t+1} = e_t, e_t \neq 0). \tag{5.1}$$

Table 5.1 shows the average repeatability of errors when up to 5 logic gates are affected by the injected faults for 16-bit and 32-bit signed and unsigned Wallace tree multipliers. For the 16-bit unsigned Wallace tree multiplier, the average repeatability of errors is higher than 0.5 when only one gate is affected. The repeatability of errors decreases as the number of affected gates increases. The signed Wallace tree multipliers based on the Booth encoding technique has smaller repeatability of errors compared to the unsigned Wallace tree multipliers. We also note that longer operand size will result in smaller error repeatability for both signed and unsigned Wallace tree multipliers. However, it should be noted that even when 5 gates are affected the repeatability of errors for the 32-bit signed Wallace tree multiplier is still around 0.05, which can be sufficient for robust and multilinear arithmetic codes to increase their error detection capabilities.

Remark 5.2.1 *For secure applications, robust and multilinear arithmetic codes can benefit from* ***design for repeatibilities****. The circuit can be designed and synthesized in such a way that the repeatability of errors is high. In this case, the error detection capabilities of robust and multilinear arithmetic codes can be drastically increased. Thereby, the security level of the system protected by these codes will be much higher. For example, reducing the average fanout of gates can result in a smaller number of possible error patterns once the fault is fixed. As a result, the repeatability of errors will increase, assuming a similar probability of fault manifestation. We also note that for linear networks consisting of only XOR gates,* $P_R = 1$ *assuming a simple stuck-at fault model.*

5.3 Robust Protection of the AES Linear Block

The Advanced Encryption Standard (AES) is one of the most used symmetric key algorithms and has been the target of numerous fault injection based attack campaigns. As in most private key algorithms, AES involves bitwise operations over small fields and the algebraic error model is most often observed and used in analysis. We use a sub-circuit of a typical round of encryption and compare architectures based on linear codes and robust codes presented in Chapter 3.

5.3.1 Hardware Architecture for Robust AES

The datapath of a typical round of AES-128 consists of four main transformations: SubBytes, ShiftRows, MixColumns and AddRoundKey. The SubBytes transformation involves two operations, inversion in $GF(2^8)$ followed by a linear affine transform. All of the transformations are defined for at most 32-bit operands and the 128-bit datapath of a round of AES can be divided into four independent and identical 32-bit data streams. For the test circuits we used the linear transformations of a 32-bit wide portion of a typical round of AES. The circuit consists of one MixColumns transformation and four affine transformations. It is completely linear and can be

implemented with 217 XOR gates.

We compare six different protection methods : linear parity, robust parity (Example 3.1.1), linear+robust parity(Example 2.3.2), partially robust $(x, (Px)^3)$ code (Example 3.3.1), Hamming code and a partially minimum distance robust code based on Vasil'ev code (Example 3.4.1). Each code protects one 32-bit linear block.

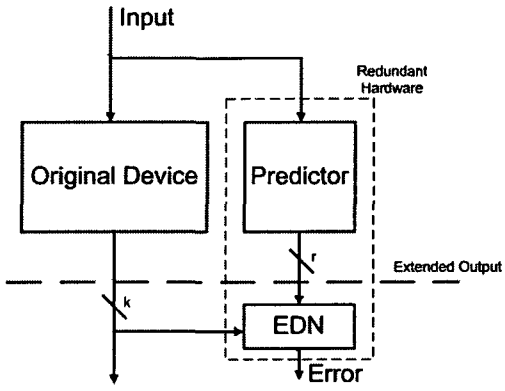


Figure 5·2: General Architecture of Secure Devices Based on Error Detecting Codes

The general architecture utilizing error detection codes to protect devices against fault analysis attacks is shown in Figure 6·1. In addition to the original device, two extra blocks, the predictor and the error detection network (EDN) are needed. The extended outputs are codewords of the error detection code. The predictor predicts the redundant outputs from the inputs of the devices. EDN is used to verify the integrity of the output data. By selecting an appropriate error detection code and implementing the corresponding predictor and EDN, the desired level of error detection capability can be achieved.

The overhead of each of the implementations are summarized in Table 1. The table lists the number of two-input gates required for each implementation and the overhead compared to the unprotected implementation. The linear parity requires very little overhead due to the parity preserving nature of the linear operations. For this sub-

Table 5.2: Hardware Overhead of Secure AES Linear Blocks Based on Different Error Detecting Codes

	predictor	EDN	overhead(%)	$max_{e\neq 0}Q(e)$	$dim(K_d)$
linear parity	31	32	30%	1	32
robust parity (Example 3.1.1)	185	32	100%	0.5	0
min. dist. robust (Example 2.3.2)	196	64	120%	0.5	0
Hamming	253	80	153%	1	32
gen. Vasil'ev (Example 3.4.1)	292	116	188%	0.5	6
$(x,(Hx)^3)$(Example 3.3.1)	432	266	322%	2^{-5}	26

circuit of AES, the parity of the inputs is equal to the parity of the outputs which results in a compact parity predictor and a 30% gate count overhead. The robust parity code requires the prediction of a nonlinear function of the output and has a larger area overhead. The implementation which combines the the linear and robust parity into one implementation requires slightly more hardware in the predictor and 32 more gates in the EDN compared to robust parity. The implementations based on Hamming codes, generalized Vasil'ev codes and $(x,(Px)^3)$ codes require much larger overhead due to the fact that more redundant bits need to be predicted. The overhead of the scheme utilizing generalized Vasil'ev code is slightly higher than Hamming implementation because it needs to compute one nonlinear redundant bit. Finally, the scheme based on $(x,(Px)^3)$ code requires more than 300% hardware overhead because the predictor and EDN needs to implement a cube operation in $GF(2^6)$ in addition to the matrix multiplication in $GF(2)$.

5.3.2 Error Detection Analysis

To illustrate the error detection characteristics of robust codes and their variants we first show results of exhaustive simulations comparing the error detection ability of codes with smaller dimensions.

Figure 5·3 shows the percentage of errors e that are masked as a function of the the error masking probability $Q(e)$ where the number of information bits is $k = 8$. Linear parity code has the largest portion of undetectable errors (50 %). For robust

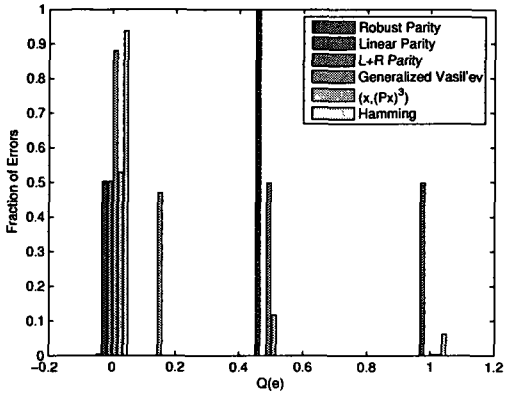


Figure 5·3: Error Distributions for Codes with $k = 8$

parity code, all errors are detectable with a probability of at least 0.5. The code with both linear and robust parity bits can detect 50% errors with probability 1 and all the others with probability 0.5. Generalized Vasil'ev code, $(x, (Px)^3)$ code and Hamming code have Hamming distance 3 and can detect all single and double errors which are most probable in practice. The first two have much smaller portion of undetectable errors than Hamming code due to their robustness. For generalized Vasil'ev code nearly 90% of errors are always detectable. $(x, (Px)^3)$ code can detect only 50% errors with probability one, but it can detect another 45% errors with probability 0.875.

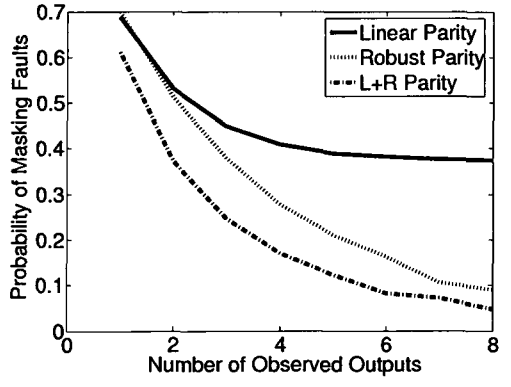


Figure 5·4: Probability of Missing Faults for Different Length of Input Sequences–Linear Parity, Robust Parity, L+R Parity ($k = 32, r = 6$)

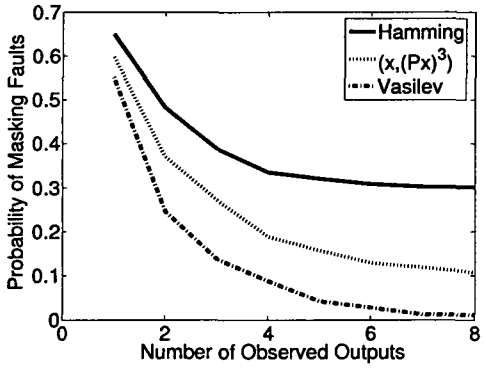


Figure 5·5: Probability of Missing Faults for Different Length of Input Sequences–Hamming, gen.Vasil'ev, $(x, (Px)^3)$ where $k = 32$

The experimental results of fault simulations for the linear sub-circuit of AES protected with the above six different codes are shown in Figure 5·4 and Figure 5·5. Single stuck-at faults were injected into the original and predictor portions of the corresponding six designs. Due to the linear function of the AES sub-circuit the faults tend to manifest themselves as repeating errors at the outputs. It is shown in (Karpovsky et al., 2007) that robust codes have better detection characteristics in channels where errors have a high laziness or probability of repeating themselves. Thereby, robust and minimum distance robust codes are expected to have better performance when faults stay for several consecutive operations.

For each of the architectures the probability of not detecting a fault at least once decreases as more outputs are observed. Due to the large kernels of the linear parity and Hamming codes and the structure of the circuit based on these codes, about 30% of single faults result in errors which are undetectable. As shown in the Figures, the probability of not detecting a fault at least once after eight messages approaches 30% for both architectures.

Robust codes have no undetectable errors and partially robust codes reduce the number of undetectable errors over linear codes. For the robust and partially robust codes the probability of not detecting a fault at least once after several messages is

much smaller than linear codes.

We note that protection methods aiming at only increasing the Hamming distance of codes do not bring big improvement for the error detection probabilities as compared with schemes based on codes with distance 1 or 2. The architecture based on Hamming code is only a little bit better than that based on linear parity code and is much worse than the one based on robust parity code in terms of fault masking probability when several consecutive outputs are observed. The reason is that most of single stuck-at faults will result in single errors or affect an odd number of output bits, which can be detected by linear parity code. If faults do manifest themselves as errors with high multiplicities, Hamming codes still do not have benefits due to the large number of undetectable errors and the disadvantage of detecting repeating errors compared with robust codes or partially robust codes. Thereby, we claim that to further increase the fault detection capability, robust codes and partially robust codes with minimum distances are better choices than linear codes with higher Hamming distances.

5.4 Robust FSMs Resilient to Advanced Fault Injection Attacks

Most of the current research on protecting cryptographic devices against fault injection attacks target at the data path of the system. On the contrary, few papers have been published on protecting the control circuit (e.g. FSM, pipeline) of the device. In (Sunar et al., 2007a), the author showed that by injecting faults into the FSM of the cryptographic device implementing the Montgomery ladder algorithm, the attacker can still reveal the secret key of the system even if the data path is properly protected. Thereby, the security of the FSMs should also be considered when designing cryptosystems resistant to fault injection attacks.

The design of reliable FSM architectures tolerant to naturally introduced errors (e.g. soft errors) are well studied in the community (Rochet et al., 1993; Krasniewski, 2008; Baranov et al., 2009). Most of these reliable FSM architectures are based on linear codes (e.g. TMR, parity prediction) and assume a specific error model where errors with small multiplicities are more probable. They cannot provide a guaranteed level of protection against fault injection attacks under strong attacker models since errors introduced by an attacker can be unpredictable.

Robust codes presented in Chapter 3 can provide equal protection against all error patterns thus eliminate the possible weakness that can be used by the attacker to break the security of the system. However, the advantage of robust codes lies on the assumption that all codewords are equi-probable. For FSMs only some of the codewords correspond to valid states. Moreover, in most of the cases the probability of valid states is not uniformly distributed. Due to these two inherent characteristics of FSMs, protection architectures based on single robust codes cannot be directly applied to build secure FSMs.

As a solution for the protection of FSMs against strong attackers, the authors in (Hammouri et al., 2009) proposed to use fingerprints generated by physically unclonable functions (PUFs) to verify the transition of the FSM when it is in operation. However, the architecture can only be applied to known-path state machines where the state transitions do not depend on the external inputs. In (Akdemir et al., 2009), a secure FSM architecture based on nonlinear functions and randomized maskings was proposed. While interesting and efficient as a countermeasure against strong attackers, the method requires a high hardware overhead.

In this section, we propose to use multi-code techniques presented in Chapter 4 to protect FSMs against fault injection attacks. The proposed architectures can provide a guaranteed level of security under strong attacker models and require less hardware

overhead than other existing secure FSM architectures described in the literature.

5.4.1 Capabilities and Goals of Attackers when Injecting Faults into FSMs

We assume that the attacker knows the detailed implementation of the secure FSM architectures. To reveal the secret information of the system, the attacker tries to force the FSM into a faulty but valid state by injecting faults into system registers or combinational networks resulting in additive errors in the system registers. Denote by x the content of a system register and e the error vector introduced by the attacker, the distorted content is $\tilde{x} = x \oplus e$, where $\oplus$ is the bit-wise XOR operation.

We further assume that the attacker cannot first read the contents from the registers and then decide how the faults will be injected during the same clock cycle. However, the attacker may be able to inject any specific error pattern. (Note that this is a stronger attacker model than the one presented in Section 5.1.) Moreover, the attacker may know the next state of the FSMs before he injects faults. This is probable for some known-path FSMs where the state transitions are not dependent upon the external inputs (Hammouri et al., 2009).

To conduct a more comprehensive comparison of different alternatives, we analyze their error detection capabilities for three different attacker models.

A1 (Weak Attacker) : The attacker injects random errors with uniform distribution.

A2 (Strong Attacker) : The attacker has high spatial fault injection resolution and is able to inject any specific error patterns. But he does not know the next state of the FSM before he injects faults.

A3 (Strongest Attacker) : The attacker has high spatial and temporal fault injection resolution. He knows the next state before he injects faults and is able to introduce any specific error patterns.

We further notice that the attacker may have different goals of conducting fault injection attacks.

G1: Force the FSM into an arbitrary faulty(but valid) state.

G2: Force the FSM into a certain faulty state as desired by the attacker.

We denote by Q_1 and Q_2 the probability that the attack is successful for the above two situations respectively. The performance of the protection architectures will be evaluated by computing Q_1 and Q_2 under all the three attacker models.

5.4.2 Secure FSM Architectures

Throughout the rest of the Section, we denote by (n, k) a binary systematic code with length n and dimension k. Let S be the set of binary vectors representing the valid states of the FSM and s an element in S. For situation $G2$, let s' be the certain state that the attacker wants to force the FSM into. Let $p(s)$ be the probability that the next state of the FSM is s, assuming the external inputs are uniformly distributed. (When treating FSMs as Markov chains, $p(s)$ is the stationary distribution of the chain.) We will only present the protection architectures for the computation of the next state functions. The computation of the output functions can be protected using similar techniques.

Architectures Based on Linear Codes

Figure 5·6 shows the general secure FSM architecture based on a (n, k) systematic code C, which consists of two registers and three combinational networks. The NSL block computes the next state vector based on the current state and the external inputs. The predictor computes the redundant bits $v \in GF(2^r)$ of the code, where $r = n - k$. The state register stores the next state vector $s \in GF(2^k)$. The check-bit register stores the redundant bits v. The non-distorted outputs of the two registers

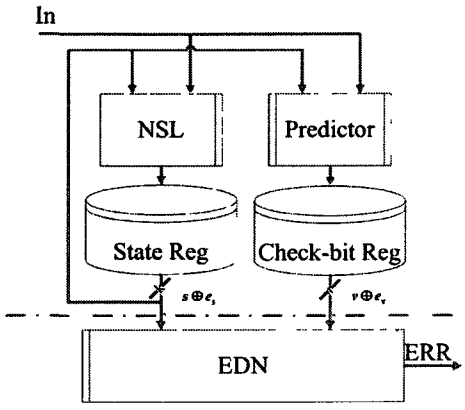


Figure 5·6: Secure FSM Architectures Based on Systematic Error Detection Codes

compose a codeword of C. Denote by e_s and e_v the additive errors occurring to the state vectors and the redundant bits respectively. At the beginning of each clock cycle, the error detection network (EDN) will verify whether $(s \oplus e_s, v \oplus e_v)$ is a codeword of C and $s \oplus e_s$ is a valid state vector. If either of the two verifications is failed, errors are detected and ERR will be asserted.

Although architectures based on linear codes can provide a satisfactory protection against most of the naturally introduced errors, when facing an attacker with advanced fault injection mechanisms, the security level of the system cannot be guaranteed.

Theorem 5.4.1 *Let $|S|$ be the number of valid states of the FSM. Let $S_{e_s} = \{s_1 \in S | \exists s_2 \in S, s_1 \oplus s_2 = e_s\}$. For architectures based on any (n, k) linear error detection code, Q_1 and Q_2 for different attacker models described in Section 5.4.1 are as stated below, where $p(s)$ is the probability that the next state of the FSM is s.*

A1: $Q_1 = \frac{|S|-1}{2^n}$, $Q_2 = \frac{1-p(s')}{2^n}$;
A2: $Q_1 = \max_{e_s}\{\sum_{s \in S_{e_s}} p(s)\}$, $Q_2 = \max_{s \neq s'}\{p(s)\}$;
A3: $Q_1 = Q_2 = 1$.

Proof Let H be the parity check matrix of the linear code and $e = (e_s, e_v), e_s \neq \mathbf{0} \in GF(2^k)$ the error introduced by the attacker. Let s be the correct output of

the NSL block (stored in the state register). The attack is partially successful iff $He = \mathbf{0} \in GF(2^r)$ and $s \oplus e_s$ is a valid state. The attack is successful iff $He = \mathbf{0}$ and $s \oplus e_s = s'$.

A1: The probability that $s \oplus e_s, e_s \neq 0$ is a valid state is $\frac{|S|-1}{2^k}$. For each e_s, there is only one e_v satisfying $He = 0$. Thereby $Q_1 = \frac{|S|-1}{2^n}$. For a given $s \neq s'$, there is only one $e = (e_s, e_v)$ satisfying $s \oplus e_s = s'$ and $He = 0$. Hence $Q_2 = \frac{1-p(s')}{2^n}$.

A2: If the attacker is able to inject specific error patterns, he can select the error which is most likely to be missed. Such an error satisfies that $e = (e_s, e_v)$ is a codeword and $\sum_{s \in S_{e_s}} p(s)$ is maximized. When this error is injected, $Q_1 = \max_{e_s}\{\sum_{s \in S_{e_s}} p(s)\}$. In order to force the FSM into a certain faulty state s', the attacker has to introduce an error $e_s = s \oplus s', s \neq s'$. If s maximizes $p(s)$ among the rest of the valid states except s', $Q_2 = \max_{s \neq s'}\{p(s)\}$.

A3: Given the fact that the attacker knows the next state s and can inject any specific error pattern e_s, he can simply inject the error $e_s = s \oplus s'$, thus $Q_1 = Q_2 = 1$.■

We next present an example of utilizing linear codes for the protection of the FSM for the Montgomery ladder algorithm, which is widely used in RSA and elliptic curve cryptosystems. The state transition diagram of the Montgomery ladder algorithm is shown in Figure 5·7. The algorithm is for the computation of $y = x^b \mathrm{mod} N$, where x is the original message and b is the m-bit secret key of the cryptosystems. After loading x and b into system registers, the FSM takes m clock cycles to finish the computation. A possible attack scenario was presented in (Sunar et al., 2007a). The authors showed that by forcing the FSM into the *DataRead* state before all the computation is completed, b can be easily revealed one bit per time. (Fore more details about the Montgomery ladder algorithm and the attack scenario, please refer to (Sunar et al., 2007a; Joye and Yen, 2003)).

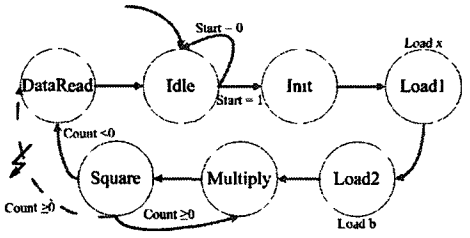


Figure 5·7: State Transition Diagram of the FSM for the Montgomery Ladder Algorithm (Sunar et al., 2007a)

The state assignment for the FSM is shown in Table 5.3.

$$S = \{001, 010, 011, 100, 101, 110, 111\}.$$

S_{e_s} can be derived from the definition in Theorem 5.4.1, e.g.

$$S_{001} = \{010, 011, 100, 101, 110, 111\}.$$

Assuming a public key size of 17-bit for RSA ($m = 17$), $p(s)$ is shown in the last column of Table 5.3.

Example 5.4.1 *The FSM for the Montgomery ladder algorithm can be protected using a* $(6, 3)$ *linear Hamming code whose parity check matrix is*

$$H = \begin{bmatrix} 100101 \\ 010110 \\ 001011 \end{bmatrix}.$$

To reveal the secret key b of the cryptosystem, the attacker tries to force the FSM

Table 5.3: State Assignment of the FSM for the Montgomery Ladder Algorithm

Valid State	State Vector	$p(s)$
Idle	001	1/39
Init	010	1/39
Load1	011	1/39
Load2	100	1/39
Multiply	101	17/39
Square	110	17/39
DataRead	111	1/39

into state 111 *(DataRead) before the computation is completed. Thereby* $s' = 111$. *Given the state assignment in Table 5.3,* $\max_{s \neq s'}\{p(s)\} = 17/39 = 0.4359$ *(s can be either* 101 *or* 110*).* $\max_{e_s}\{\sum_{s \in S_{e_s}} p(s)\} = 38/39 = 0.9744$. e_s *can be any vector in* $\{001, 010, 011, 100, 111\}$. Q_1 *and* Q_2 *under different attacker models are shown in Table 5.4. Under attacker models* **A2** *and* **A3**, Q_1 *is close to* 1 *and* Q_2 *is at least* 0.4359. *Obviously, FSM protection architectures based on single linear codes cannot provide enough protection in these situations.*

Remark 5.4.1 • *We note that for the above FSM some mis-transitions of the states may not reveal secret information of the cryptosystem. However, to compare different alternatives for more general situations, we still use the definition of* Q_1 *and* Q_2 *given in Section 5.4.1.*

• *Since m clock cycles are required to finish the computation after x and b are loaded, an extra counter is needed to store the number of passed clock cycles (see Count in Figure 5.7). This counter should be protected using similar techniques presented in this paper.*

From Theorem 5.4.1 it is clear that for FSM protection architectures based on single linear codes, Q_1 and Q_2 do not depend on the code type. When the attacker can only inject random errors (**A1**), Q_1 and Q_2 are affected by n and $p(s)$. When the attacker is able to inject specific error patterns (**A2**), Q_1 and Q_2 are affected by $p(s)$ and S_{e_s}. In general, to achieve a higher security level of FSMs, $\max_{s_1 \neq s_2}\{p(s_1) - p(s_2)\}$ should be as small as possible. If the attacker also knows the next sate of the FSM (**A3**), any protection architectures based on single linear codes stand no chance.

In the next section, we will show a possible solution for the protection of FSMs against strong attackers, which is based on multi-code techniques.

Architectures Based on Multi-Code Techniques

The general secure FSM architecture based on multi-code techniques is shown in Figure 5.4.2. At each clock cycle, a selection signal $R \in GF(2^{r_L})$ is generated by a (pseudo) random number generator, which is integrated in most of the cryptographic

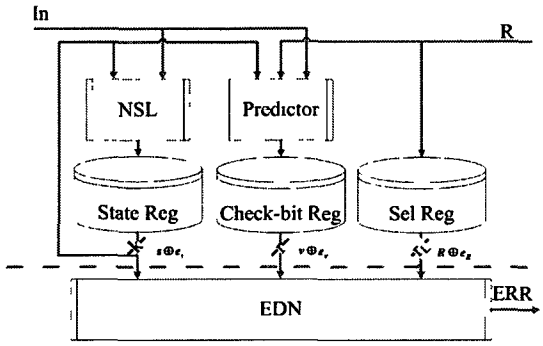


Figure 5·8: Secure FSM Architectures Based on Multi-code Techniques

devices. Based on the value of R, the predictor selects a code from $L, L \leq 2^{r_L}$ different codes to encode the next state vector s. R is stored in a separate register and is used to verify s at the beginning of the next clock cycle.

Remark 5.4.2 *As in most papers on protecting the cryptographic devices, e.g. see (Kulikowski et al., 2008b; Malkin et al., 2006; Akdemir et al., 2011), we assume that countermeasures are implemented in the cryptographic devices to prevent the attacker from tampering with the clock signals or the random number generators.*

Given a set of linear codes, if the intersection of any two of these codes contains only the all-zero vector, we say that these codes are **non-overlapping**. The next theorem shows that by randomly selecting one code from a set of non-overlapping linear codes for each encoding and decoding operation, we can effectively reduce the chance for the attacker to conduct a successful fault injection attack.

Theorem 5.4.2 *Let $C_1, C_2, \cdots, C_L$ be L different (n,k) linear codes satisfying $C_i \cap C_j = \mathbf{0} \in GF(2^n), i \neq j, 1 \leq i,j \leq L \leq 2^{r_L}$. Let $R \in GF(2^{r_L})$ be the randomly generated selection signal with uniform distribution and $e_R \in GF(2^{r_L})$ be the additive error in the register storing the value of R. Let Γ be the set of all valid selection signals, $|\Gamma| = L$. Assume that for every nonzero e_R, there is at most one pair of R, R' satisfying $R, R' \in \Gamma, R \oplus R' = e_R$. If we randomly select C_i for each encoding and decoding operation, Q_1 and Q_2 for different attacker models are as stated below.*

A1: $Q_1 = \frac{L(|S|-1)}{2^{n+r_L}}$, $Q_2 = \frac{L(1-p(s'))}{2^{n+r_L}}$;

A2: $Q_1 = \frac{1}{L}\max_{e_s}\{\sum_{s\in S_{e_s}} p(s)\}$,
$Q_2 = \frac{1}{L}\max_{s\neq s'}\{p(s)\}$;
A3: $Q_1 = Q_2 = \frac{1}{L}$.

Proof Denote by H_i the parity check matrix of the i_{th} linear code. Let (e_s, e_v) be the error injected to the codeword and e_R be the error injected to the register storing the selection signal R. $G1$ is achieved iff $H_{R\oplus e_R}(s\oplus e_s, v\oplus e_v) = 0$ and $s\oplus e_s$ is a valid state, $e_s \neq 0$. $G2$ is achieved iff $H_{R\oplus e_R}(s\oplus e_s, v\oplus e_v) = 0$ and $s\oplus e_s = s'$.

A1: The probability that $e\oplus e_s$ is a valid state is $\frac{|S|-1}{2^k}$. The probability that $s\oplus e_s = s', e_s \neq 0$ is $\frac{1-p(s')}{2^k}$. For each e_s, there are L pairs of e_R and e_v satisfying $H_{R\oplus e_R}(s\oplus e_s, v\oplus e_v) = 0$. Thereby, $Q_1 = \frac{L(|S|-1)}{2^{n+r_L}}$, $Q_2 = \frac{L(1-p(s'))}{2^{n+r_L}}$.

A2: The maximum probability that $s\oplus e_s$ is a valid state for a given e_s is

$$\max_{e_s}\{\sum_{s\in S_{e_s}} p(s)\}.$$

The maximum probability that $s\oplus e_s = s'$ is $\max_{s\neq s'}\{p(s)\}$. If $e_R = 0$, $Q_1 = \frac{1}{L}\max_{e_s}\{\sum_{s\in S_{e_s}} p(s)\}$, $Q_2 = \frac{1}{L}\max_{s\neq s'}\{p(s)\}$. If $e_R \neq 0$, under the assumption of Theorem 5.4.2, there exists only one $R'\in\Gamma$ such that $R\oplus R' = e_R$. Rewrite H_i into the standard form $H_i = (I, P_i)$, where I is the $r\times r$ identity matrix and P_i is the $r\times k$ encoding matrix of C_i. Suppose $e = (e_s, e_v)$ converts an encoded state s in C_R into another encoded state s' in $C_{R'}$. Then $e_v = (P_R\oplus P_{R'})s\oplus P_{R'}e_s$. (The same $e = (e_s, e_v)$ can also convert s' in $C_{R'}$ back into s in C_R). For each $e = (e_s, e_v)$, there is at most one s satisfying the above equation. If the attacker inject nonzero e_R, it is easy to show that Q_1 and Q_2 will be no larger than the case when $e_R = 0$. Thereby, in this situation $Q_1 = \frac{1}{L}\max_{e_s}\sum_{s\in S_{e_s}} p(s)$, $Q_2 = \frac{1}{L}\max_{s\neq s'}\{p(s)\}$,

A3: In the last situation, following similar analysis we can show that $Q_1 = Q_2 = \frac{1}{L}$.

■

Remark 5.4.3 *Γ is a 2-robust code (Chapter 2), i.e. each nonzero error is masked by at most two codewords of the code.*

Compared to the architectures based on single linear codes, architectures based on multilinear codes reduce Q_1 and Q_2 by a factor of L under attacker models **A2** and **A3** thus largely increase the security level of the system.

Several constructions of non-overlapping multilinear algebraic codes have been presented in Chapter 4.

Example 5.4.2 *In this example, we use four non-overlapping $(6,3)$ linear codes $C_i, 1 \leq i \leq 4$ to protect the FSM for the Montgomery ladder algorithm. Let*

$$P_1 = \begin{bmatrix} 101 \\ 110 \\ 011 \end{bmatrix}, P_2 = P_1 \oplus \begin{bmatrix} 100 \\ 010 \\ 001 \end{bmatrix}$$

$$P_3 = P_1 \oplus \begin{bmatrix} 010 \\ 001 \\ 110 \end{bmatrix}, P_4 = P_1 \oplus \begin{bmatrix} 001 \\ 101 \\ 111 \end{bmatrix}$$

be the encoding matrices of the codes. It is easy to verify that $P_{ij}, 1 \leq i,j \leq 4$ has rank 3 and the intersection of any two of these codes contains only the all-zero vector. Let $r_L = 3$ and $001, 010, 100, 011$ be the selection signal for $C_i, 1 \leq i \leq 4$ respectively. ($\Gamma = \{001, 010, 100, 011\}$.) For every $e_R \neq 0$, there is at most one pair of $R, R' \in \Gamma$ satisfying $R \oplus R' = e_R$. Given the state assignment in Table 5.3, if we randomly select C_i for every encoding and decoding operation, Q_1 and Q_2 under different attacker models are shown in Table 5.4. Compared to Example 5.4.1, the described method can reduce Q_1 and Q_2 by a factor of four under attacker models **A2** *and* **A3**. *A higher security level can be achieved by randomly selecting from more linear codes (increase k and r if necessary).*

Similar multi-code techniques can also be applied to nonlinear robust codes. We next describe a secure FSM architecture based on **multirobust** codes, which has highly regular structures for the encoder and the decoder and results in comparable hardware overhead to architectures based on multilinear codes. The error detection

capability of the proposed architecture will be analyzed and compared to other alternatives.

Theorem 5.4.3 *Let* $C_i = \{(s,v)|s \cdot v = i-1\}, 1 \leq i \leq 2^k$, *where* $s, v \in GF(2^k)$ *and* $\cdot$ *is the multiplication in* $GF(2^k)$. *Assume that* $\mathbf{0} \in GF(2^k)$ *is not a valid state* ($\mathbf{0} \notin S$). *If we randomly select* $C_i, 1 \leq i \leq 2^k$ *for each encoding and decoding operation* ($L = 2^k$), Q_1 *and* Q_2 *under different attacker models are as stated below.*

A1: $Q_1 = \frac{|S|-1}{2^{2k}}$, $Q_2 = \frac{1-p(s')}{2^{2k}}$;
A2: $Q_1 = \frac{1}{2^k} \max_{e_s}\{\sum_{s \in S_{e_s}} p(s)\}$,
$Q_2 = \frac{1}{2^k} \max_{s \neq s'}\{p(s)\}$;
A3: $Q_1 = Q_2 = \frac{1}{2^k}$.

Proof An error $e = (e_s, e_v)$ will be missed iff $(s \oplus e_s) \cdot (v \oplus e_v) = R \oplus e_R$. Since $v = \frac{R}{s}, s \neq 0$, the above equation can be re-written as $e_v \cdot s^2 \oplus (e_s \cdot e_v \oplus e_R) \cdot s \oplus e_s \cdot R = \mathbf{0}$.

A1: The probability that $e_s \oplus s$ is a valid state is $\frac{|S|-1}{2^k}$. The probability that $e_s \oplus s = s', s \neq s'$ is $\frac{1-p(s')}{2^k}$. For each $e = (e_s, e_v)$, there is only one e_R satisfying the above equation. Thereby $Q_1 = \frac{|S|-1}{2^{2k}}$. $Q_2 = \frac{1-p(s')}{2^{2k}}$.

A2: The maximum probability that $s \oplus e_s$ is a valid state for a given e_s in this case is $\max_{e_s}\{\sum_{s \in S_{e_s}} p(s)\}$. The maximum probability that $s \oplus e_s = s'$ is $\max_{s \neq s'}\{p(s)\}$. For any fixed $e_s \neq 0, e_v$ and e_R, there is only one R for each s satisfying the error masking equation. Hence $Q_1 = \frac{1}{2^k} \max_{e_s}\{\sum_{s \in S_{e_s}} p(s)\}$. $Q_2 = \frac{1}{2^k} \max_{s \neq s'}\{p(s)\}$.

A3: Following similar analysis we can show that Q_1 and Q_2 are at most $\frac{1}{2^k}$. ■

For a given number of redundant bits $r = k$, the architecture based on Theorem 5.4.3 has the maximum L thus can minimize Q_1 and Q_2 under attacker models **A2** and **A3**. The encoder of the proposed multirobust codes mainly contains an inverse operation and a multiplication in $GF(2^k)$. The EDN of the multirobust codes requires only one multiplication in $GF(2^k)$. Architectures based on Theorem 5.4.3 require

less overhead than other existing secure FSM architectures utilizing nonlinear robust codes. (For instance, the architecture proposed in (Akdemir et al., 2009) requires at least 4 cubings and 2 multiplications in $GF(2^k)$).

Example 5.4.3 *The FSM of the Montgomery ladder algorithm can also be protected using eight* (6.3) *non-overlapping nonlinear codes as described in Theorem 5.4.3. We still use the state assignment in Table 5.3. (Note that the all-zero vector is not a valid state.) If we randomly select from* $C_i = \{(s, v)|s \cdot v = i - 1\}, 1 \leq i \leq 8$ *for every encoding and decoding operation,* Q_1 *and* Q_2 *will be reduced by a factor of* 8 *under attacker models* **A2** *and* **A3** *compared to architectures based on linear codes (Table 5.4). Similar to protection methods based on multilinear codes, higher security level can be achieved by randomly selecting from a larger set of codes (increase* k *and* r*).*

Table 5.4: Q_1 and Q_2 of Secure FSM Architectures for the Montgomery Ladder Algorithm

attacker models	Code	Q_1	Q_2
$A1$	Linear (Exp. 5.4.1)	0.0938	0.0152
	Multilinear (Exp. 5.4.2)	0.0469	0.0076
	Multirobust (Exp. 5.4.3)	0.0938	0.0938
$A2$	Linear (Exp. 5.4.1)	0.9744	0.4359
	Multilinear (Exp. 5.4.2)	0.2436	0.1090
	Multirobust (Exp. 5.4.3)	0.1218	0.0545
$A3$	Linear (Exp. 5.4.1)	1	1
	Multilinear (Exp. 5.4.2)	0.2500	0.2500
	Multirobust (Exp. 5.4.3)	0.125	0.125

The error detection capabilities of different secure FSM architectures (Example 5.4.1, 5.4.2, 5.4.3) for the Montgomery ladder algorithm are shown in Table 5.4. All the data are verified via simulation in MATLAB. When the attacker is able to inject specific error patterns (attacker models $A2$ and $A3$), architectures based on multilinear codes and multirobust codes can reduce Q_1 and Q_2 by a factor of L and 2^k respectively compared to architectures based on single linear codes (in our case $L = 4$, $2^k = 8$). Thereby, under the assumption of a strong attacker model, the proposed methods can provide a guaranteed level of security which cannot be achieved using architectures based on single linear codes.

Table 5.5: Structure, Hardware and Power Consumption Overhead of Encoders and EDN of Different Alternatives

code	Structure		Overhead	
	Linear	Nonlinear	Hardware	Power
Linear	$\surd$	–	44.37%	37.66%
Multilinear	$\surd$	–	129.0%	94.13%
Multirobust	–	$\surd$	119.5%	133.57%

To estimate the overhead, the three designs were modeled in VERILOG, synthesized using Cadence RTL Compiler and placed & routed using Cadence Encounter based on Nangate 45nm open cell library (Nangate, 2011). The structure and the overhead of the encoder and EDN of different alternatives are shown in Table 5.5. All the structures are linear except for the encoder and EDN of architectures based on multirobust codes. Generally speaking, the implementation of nonlinear operations requires more hardware overhead than the implementation of linear operations. However, due to the regular structure of the codes, the secure FSM architecture based on multirobust codes has comparable hardware overhead to architectures based on multilinear codes for small k. For the proposed architectures based on multi-code techniques, the security level of the system can be increased by increasing L and k. When $k = 32$, Q_1 and Q_2 for architectures based on multirobust codes can be as small as 2^{-32}.

5.5 Secure Multipliers Based on Multilinear Arithmetic Codes

The multiplier is a basic block in many public key cryptographic devices. Due to its arithmetic nature of the operations, arithmetic error model is most often used for such devices. We assume that faults manifest as additive arithmetic errors at the output of the multiplier and the predictor [1]. The error is in the format of $e =$

[1]The term *predictor* is used in this context to refer to the circuit that computes the redundant bits of the output of the operation directly from the inputs. In our case the predictor computes the redundant bits of the multiplication result.

$(e_x, e_y), e_x \in Z_{2^k}, e_y \in Z_{2^r}$, where k is the number of information bits, r is the number of redundant bits and Z_{2^k} is the additive group of integers $\{0, 1, \cdots, 2^k - 1\}$. In this section, we analyze and compare the hardware overhead, the number of bad errors (see Section 4.2) and the fault detection capabilities for architectures protected by linear, multilinear (see Section 4.2.1) and robust arithmetic codes (Kulikowski et al., 2008b).

5.5.1 Hardware Overhead

The general architecture of multipliers protected by block codes contains three parts: the original multiplier, the predictor that generates the redundant bits of the code and the error detection network (EDN). The detailed architectures for secure multipliers protected by linear and multilinear arithmetic codes are shown in Figure 5·9. For the architecture based on $(x, |x^2|_p)$ partially robust arithmetic codes, please refer to (Kulikowski et al., 2008b).

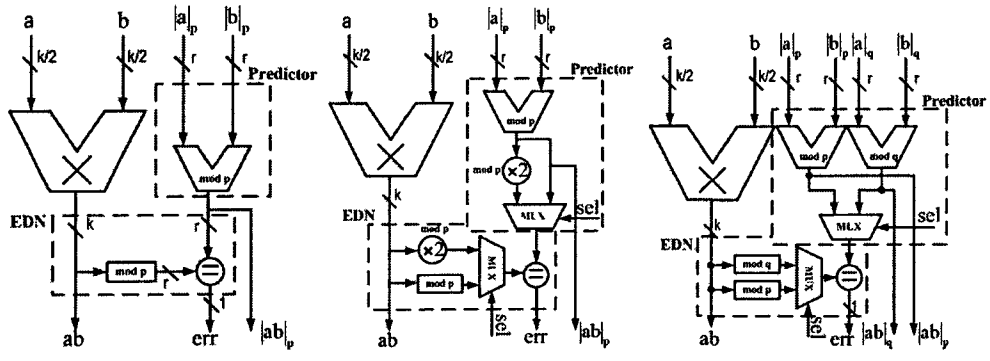


Figure 5·9: Hardware architectures for multipliers protected by (a) linear arithmetic codes, (b) $[|x|_p, |2x|_p]$ multilinear codes and (c) Multi-modulii multilinear codes

The predictor for the linear arithmetic codes contains one multiplier in Z_p. Except for the r-bit comparator, the only operation implemented in the error detection network is a modulo p operation. The hardware overhead mainly comes from the

r-bit modulo p multiplier ($r = \lceil \log_2(p) \rceil$), whose complexity is of the order of $O(r^2)$, and the modulo p operation in EDN, whose complexity is $O(k)$. (k is the number of information bits).

Compared with architectures based on linear arithmetic codes, the architecture utilizing $[|x|_p, |2x|_p]$ multilinear codes only needs one extra r-bit multiplexer and one extra multiply-by-2 operation in Z_p for both the predictor and the EDN. Multiply-by-2 operation is equal to shifting the operands by 1 bit, which is trivial in terms of the hardware overhead. We assume that the complexity of a r-bit multiplexer is in general of the order of $O(r)$. Thereby this architecture has comparable hardware overhead to the one for linear arithmetic codes.

The protection architecture based on multi-modulii multilinear codes needs one more multiplier in Z_q for the predictor. When $p << 2^k$, which is often the case in real life, q should be selected as the largest prime number that is smaller than p if we want to minimize the number of bad errors. A multiplier in Z_q will have about the same hardware complexity as the multiplier in Z_p and this will double the overhead for the predictor. However, we claim that a merged design of the two multipliers for the predictor should be implemented. First, from the security point of view, separate redundant data path may be used by attackers to derive the secret information of the devices, e.g. the attacker can inject faults into one redundant path of the device which will never influence the other. A merged design can effectively solve the problem because most of the faults injected into the redundant part of the device will affect the generation of both $|x|_p$ and $|x|_q$. Second, the hardware overhead for the predictor will be reduced if we merge the design of the two multipliers. A more aggressive approach is to design the original multiplier and the predictor of the code together as discussed in Section 4.2.1.

Remark 5.5.1 *There is a tradeoff between the error detection capabilities and the*

hardware overhead when we select p and q. Specialized p and q can significantly reduce the hardware complexity of the modulo operation, e.g. using Mersenne primes (Tahir et al., 1995).

To compare the hardware area overhead, we modeled 16-bit Wallace tree multipliers protected by linear, partially robust and multilinear codes in Verilog and synthesized them in RTL design compiler using Nangate 45nm technology (Nangate, 2011). The area comparison was based on synthesized results. The results are shown in Table 5.6. We selected p to be 31. For multi-modulii multilinear codes, q was selected to be 29. The percentage overhead was computed by dividing the estimated gate area of the predictor and EDN by the estimated area of the Wallace tree multiplier. As expected, secure multipliers based on $[|x|_p, |2x|_p]$ codes have similar overhead to architectures based on linear arithmetic codes. Architectures based on multi-modulii codes require the largest overhead, which is around 50%. The benefit of these codes is that they have the best error detection capabilities against errors in the format of $e = (e_x, 0)$. In Section 5.5.2, we will show that this characteristics of multi-modulii codes will make them the best alternative against fault-injection attacks when the design of the predictor is separated from the original multiplier. Moreover, the hardware overhead of architectures based on multi-modulii codes will be drastically reduced if we select q to be a Mersenne prime. In fact, 50% overhead is still much smaller than overheads for architectures based on robust arithmetic codes, which is around $200\% - 400\%$ (Gaubatz et al., 2006; Kulikowski et al., 2008b).

5.5.2 Experimental Results on Comparison of Error and Fault Detection Capabilities for Linear, Partially Robust and Multilinear Arithmetic Codes

To demonstrate the advantages of multilinear codes and partially robust codes over linear codes for building secure multipliers against fault-injection attacks, we con-

Table 5.6: Hardware area overhead for architectures based on linear, multilinear and partially robust arithmetic codes ($k = 32, r = 5, p = 31, q = 29$)

Code	Predictor	EDN	Total
Linear Arithmetic Codes	9.76%	10.68%	20.44%
$[\|x\|_p, \|2x\|_p]$ Multilinear Codes	10.37%	11.75%	22.12%
Multi-modulii Multilinear Codes	16.14%	37.57%	53.71%
$(x, \|x^2\|_p)$ Partially Robust Arithmetic Codes	14.60%	14.71%	29.31%

ducted simulations to analyze and compare the number of bad errors and the fault detection capabilities of the four alternatives presented in the last section. For all the simulations, we assumed the operands of the multipliers are 16 bits. Each code has 32 information bits ($k = 32$). p and q were selected to be 31 and 29 respectively.

Number of Bad Errors

In this simulation, we randomly generate 5000 non-zero errors $e = (e_x, e_y)$. For each e, we randomly select one million messages in $Z_{2^{32}}$ and encode them using linear, $[|x|_p, |2x|_p]$, multi-modulii and $(x, |x^2|_p)$ partially robust arithmetic codes. The distorted codewords $\tilde{c} = (\tilde{x}, \tilde{y}) = (|x + e_x|_{2^k}, |y + e_y|_{2^r})$ are decoded by the error detection network. The number of codewords masking each error is recorded. The distribution of error masking probabilities of the 5000 non-zero errors is shown in Table 5.7. Most of the errors are masked with a probability of less than 10% for all the alternatives. Linear arithmetic codes have 149 bad errors which are masked by a probability of at least 0.5. The numbers of bad errors for $[|x|_p, |2x|_p]$, multi-modulii and $(x, |x^2|_p)$ codes are similar and are much smaller than that of the linear arithmetic codes, which can result in better fault detection capabilities assuming repeating errors as a result of a slow fault-injection mechanism (fault stays for several consecutive clock cycles). Compared to multilinear codes, $(x, |x^2|_p)$ codes have much less errors that are masked by a probability of more than 10%. However, we will show later in this section

that $(x, |x^2|_p)$ codes actually have the worst error detection capabilities of errors in the format of $e = (e_x, 0)$ and is only suitable for designs where the multiplier and the predictor are synthesized together. Moreover, $(x, |x^2|_p)$ codes have larger overhead than $[|x|_p, |2x|_p]$ multilinear codes. The disadvantage of overhead for $(x, |x^2|_p)$ will become more significant as k increases.

Table 5.7: Error masking probability distributions for secure multipliers based on linear, multilinear and partially robust arithmetic codes $(k = 32, r = 5, p = 31, q = 29)$

Code	$< 10\%$	$10\% - 20\%$	$20\% - 30\%$	$30\% - 40\%$	$40\% - 50\%$	$\geq 50\%$ (Bad Errors)
linear	4656	59	51	46	39	149 (29.8%)
$[\|x\|_p, \|2x\|_p]$	4492	182	142	107	74	3 (0.06%)
Multi-modulii	4444	196	140	118	97	5 (0.1%)
$(x, \|x^2\|_p)$	4996	0	0	0	0	4 (0.08%)

Fault Detection Capabilities When Both the Multiplier and the Predictor are Affected by Faults

Suppose both the original multiplier and the predictor are affected by the injected faults, which manifest as a non-zero error $e = (e_x, e_y)$ at the output of the device. Assume that each multiplication is completed in one clock cycle and the same error e stays for T consecutive clock cycles (slow fault-injection mechanism). If e is detected at least once among the T clock cycles, we say that e is detected. Otherwise e is masked. In this simulation, we randomly select 10 millions possible error patterns e and assume that e may stay up to 3 clock cycles. The average error masking probabilities of e for the four presented alternatives are shown in Table 5.8. All codes have similar error detection capabilities when e stays for only one clock cycle. However, when $T = 2$, the error masking probabilities of $[|x|_p, |2x|_p]$ and multi-modulii codes are already nearly half of that of linear arithmetic codes. As T increases, the advantage of $[|x|_p, |2x|_p]$ and multi-modulii codes become more significant. As

expected, when both the original multiplier and the predictor are affected by the injected faults, $(x, |x^2|_p)$ codes have the best error and fault detection capabilities among the four alternatives.

Fault-Injection Simulations For the Case When Only the Original Multiplier is Affected by Faults

Suppose the design of the multiplier and the predictor is separated and the attacker injects faults only to the original multiplier. In order to analyze the fault detection capabilities, we conducted gate-level fault-injection simulations in C++ on 16-bit secure Wallace tree multipliers protected by different alternatives. The gate level netlist is derived from Verilog models. Each gate may have stuck-at-0 or stuck-at-1 faults. We assume that 2 to 4 gates ($2 \leq N \leq 4$) may be affected by the injected faults and the faults stay for up to 3 consecutive clock cycles ($T \leq 3$). At each clock cycle, a new pair of operands are randomly generated and multiplied. If the manifested error is detected for at least one clock cycle, we say that the fault is detected.

Table 5.8: Fault masking probabilties when both the original multiplier and the predictor are affected ($k = 32, r = 5, p = 31, q = 29$)

T	Linear	$[\|x\|_p, \|2x\|_p]$	Multi-modulii	$(x, \|x^2\|_p)$
T=1	3.12%	3.12%	3.12%	3.12%
T=2	1.81%	0.95%	0.95%	0.10%
T=3	1.25%	0.38%	0.35%	0.003%

* T is the number of clock cycles that a fault stays.

Table 5.9 summarizes the fault masking probabilities for all combinations of N and T. When a certain number N of gates are affected (N is fixed), larger T will result in smaller fault masking probabilities. When $T = 1$, the fault masking probabilities increase as N increases. However, when $T > 2$, the fault masking probabilities will drop as N increases. This is because for larger N, errors are more probable to manifest as different non-zero errors at the output of the device. For smaller N, it is more

likely that even if the fault stays for several consecutive clock cycles, it only manifests in one clock cycle. In this case, the fault detection capabilities will not increase.

Table 5.9: Fault masking probabilties when only the original device is affected ($k = 32, r = 5, p = 31, q = 29$)

	Linear			$[\|x\|_p, \|2x\|_p]$			Multi-modulii			$(x, \|x^2\|_p)$		
T	N=2	N=3	N=4	N=2	N=3	N=4	N=2	N=3	N=4	N=2	N=3	N=4
1	2.7%	3.5%	3.5%	2.7%	3.5%	3.5%	1.6%	2.4%	2.8%	5.7%	6.5%	6.5%
2	1.0%	0.78%	0.47%	1.0%	0.78%	0.47%	0.5%	0.43%	0.29%	2.1%	1.5%	1.0%
3	0.34%	0.16%	0.06%	0.34%	0.16	% 0.06%	0.14%	0.07%	0.03%	0.71%	0.31%	0.14%

* T is the number of clock cycles that a fault stays. N is the number of gates that a fault affects.

When only the original multiplier is affected, multi-modulii codes have the best error detection capabilities. When $T = 2$, the fault masking probabilities of multi-modulii codes are nearly half of the fault masking probabilities of linear and $[|x|_p, |2x|_p]$ arithmetic codes with the same N. The advantage of multi-modulii codes becomes larger as T increases.

Linear arithmetic codes and $[|x|_p, |2x]]_p$ codes have the same error detection capabilities for errors in the format of $e = (e_x, 0)$. The reason is when $e_y = 0$, the error masking equations for $(x, |x|_p)$ and $(x, |2x|_p)$ codes are $|e_x|_p = 0, |2e_x|_p = 0$ and $|e_x - 2^k|_p = 0, |2(e_x - 2^k)|_p = 0$ depending on the ranges of $x + e_x$. Obviously, $|e_x|_p = 0$ is equivalently to $|2e_x|_p = 0$ and $|e_x - 2^k|_p = 0$ is equivalent to $|2(e_x - 2^k)|_p = 0$. Thereby, $e = (e_x, 0)$ is masked by $[|x|_p, |2x|_p]$ codes if and only if it is masked by linear arithmetic codes.

$(x, |x^2|_p)$ codes have the worst error detection capabilities for errors in the format of $e = (e_x, 0)$ among the four alternatives. When $e_y = 0$, the error masking equation for $(x, |x^2|_p)$ is $|2e_x x + e_x^2|_p = 0$ and $|2(e_x - 2^k) + (e_x - 2^k)^2|_p = 0$ for different ranges of $x + e_x$. When $|e_x| = 0$ or $|e_x - 2^k|_p = 0$, $|2e_x x + e_x^2|_p = 0$ or $|2(e_x - 2^k) + (e_x - 2^k)^2|_p = 0$ is always true. But the inverse statement is incorrect. Thereby, $(x, |x^2|_p)$ will mask more errors in the format of $e = (e_x, 0)$ than linear and $[|x|_p, |2x|_p]$ arithmetic codes.

Selection of Arithmetic Codes for Secure Multipliers

From the above analysis, linear arithmetic codes have a lot of bad errors – errors masked with a probability of at least 0.5 – which may compromise the security level of the system. $[|x|_p, |2x|_p]$, multi-modulii and $(x, |x^2|_p)$ codes have a smaller number of bad errors than linear arithmetic codes (Table 4.6 and 5.7). $[|x|_p, |2x|_p]$ and $(x, |x^2|_p)$ are more suitable for designs where the original multipliers and the predictors are synthesized together. $(x, |x^2|_p)$ have better fault and error detection capabilities while $[|x|_p, |2x|_p]$ require less hardware overhead. The selection of these two codes depends on specific applications. When the designs of the multiplier and the predictor are separated and only the multiplier is affected by the injected faults, $[|x|_p, |2x|_p]$ and $(x, |x^2|_p)$ are no better than linear arithmetic codes. In this case, we should select multi-modulii codes which have the best detection capabilities against errors in the format of $e = (e_x, 0)$.

5.6 Summary

In this Chapter we presented three case studies of using robust and multilinear codes to build secure cryptographic devices resilient to fault injection attacks.

The first case study is the protection of the AES linear block using robust codes, partially robust codes and minimum distance robust and partially robust codes. The comparison of different protection methods shows that there is a large difference in the hardware overhead depending on the robustness of the architecture. The more robust the architecture is, the more hardware overhead is necessary. While robust protection can be implemented efficiently for some highly nonlinear circuits such as SBoxes, fully robust architectures for more general circuits require an overhead of more than 100%. Partially robust codes that preserve some linear structures allow for a better minimization of the predictor and have an overhead of the order of 40%-75%.

Robust architectures do offer an advantage against an unpredictable fault attacker since the number of undetectable is greatly reduced with the nonlinear encoding. It is important to emphasize that in terms of the number of undetectable errors, linear architecture cannot reach the protection of the robust architectures regardless of the number of redundant bits and the hardware overhead added to the device. For attacks where faults of low multiplicities are known to be most likely, robust codes and partially robust codes with minimum distances resulted in better fault detection than linear codes with even higher Hamming distances.

The second case study is the design of robust FSMs based on multilinear algebraic codes. We proved that if the attacker is able to inject specific error patterns, randomly selecting among L codes for each encoding and decoding operation can reduce the chance for the attacker to conduct a successful attack by a factor of L compared to architectures based on linear codes. The proposed techniques were utilized to protect the FSM of the Montgomery ladder algorithm, which can reduce the chance for the attacker to conduct a successful attack by a factor of up to 8 assuming $k = r = 3$. The hardware overheads of the proposed architectures are $120\% - 130\%$ and are less than other secure FSM architectures (Sunar et al., 2007a) based on nonlinear codes which are resistant to strong attackers. The security level of systems protected by multi-code techniques can be further improved by increasing L.

The last case study is related to secure multipliers – a commonly used block in public-key cryptographic devices – using multilinear arithmetic codes. The hardware overhead and the error and fault detection capabilities of secure multipliers based on multilinear codes are analyzed and compared to those based on linear and partially robust arithmetic codes. Simulation results show that multilinear and partially robust arithmetic codes have smaller number of bad errors (errors masked by a probability of at least 0.5) and can provide better protection than linear arithmetic codes assuming

a slow fault-injection mechanism. The proposed codes do not imply any limitations on the types of errors at the output of the protected device, e.g. the multiplicities of the errors do not have to be small. $[|x|_p, |2x|_p]$ codes have similar overhead to linear arithmetic codes with the same number of redundant bits. $(x, |x^2|_p)$ and multi-modulii codes have slightly higher overhead than linear arithmetic codes. But the overhead is at most around 50% and is much smaller than the overhead of architectures based on robust arithmetic codes, which is around $200\% - 400\%$. If the designs of the predictor and the original multiplier are separated and the injected faults affect only the multiplier, multi-modulii code is the best alternative. In this case, the fault masking probability of architectures based on multi-modulii codes is almost twice smaller than architectures based on the other codes when the fault stays for only one clock cycle. The advantage of multi-modulii codes will become even more significant as the fault and the resulting error pattern stays longer. If the faults affect both the multiplier and the predictor, $(x, |x^2|_p)$ codes have the best fault detection capabilities. $[|x|_p, |2x|_p]$ code has similar performance to multi-modulii codes and require the least hardware overhead among multilinear and partially robust arithmetic codes. The selection of codes depends on specific applications.

Chapter 6

Algebraic Manipulation Detection Codes and Their Applications

Countermeasures against fault injection attacks based on error detecting codes assume that the attacker cannot simultaneously control the fault-free outputs of a device-under-attack and the non-zero error patterns. For advanced attackers who are able to control both of the above two aspects, traditional protections can be easily compromised. In this Chapter, we propose optimal **algebraic manipulation detection (AMD)** codes based on the nonlinear encoding functions and the true random number generator. The proposed codes can provide a guaranteed high error detecting probability even if the attacker can fully control the fault-free outputs of a device-under-attack as well as the non-zero error patterns. As a case study, we present the protection architectures based on AMD codes for multipliers in Galois fields used for the elliptic curve cryptography. The results show that the proposed architecture can provide a very low error masking probability at the cost of a reasonable area overhead. The protected multiplier has no latency penalty when the predictor is pipelined.

6.1 Strongly Secure Cryptographic Devices

Robust codes (Kulikowski et al., 2008b; Wang et al., 2010a) are designed to provide a guaranteed level of detection against all error types and classes, assuming the attacker

cannot control the fault-free outputs of the cryptographic devices. These codes can be easily compromised when the above assumption is not valid.

Example 6.1.1 *Suppose the 32-bit device is protected by a robust duplication code* $C = \{y, f(y)\}$, *where* $y, f(y) \in GF(2^{32})$, $f(y) = y^3$ *and all operations are in* $GF(2^{32})$. *It is easy to prove that any non-zero error e will be masked by at most two codewords (Karpovsky and Taubin, 2004), i.e. for any non-zero error* $e = (e_y, e_f)$ *there exist at most two vectors* $y_1, y_2 \in GF(2^{32})$ *such that* $(y_1 \oplus e_y)^3 = y_1^3 \oplus e_f$ *and* $(y_2 \oplus e_y)^3 = y_2^3 \oplus e_f$. *Assume that an attacker cannot control the fault-free outputs y during attacks and the outputs of the original device are uniformly distributed, then the probability that the attacker conducts a successful attack* ($(e = (e_f, e_f))$ *is not detected) is at most* 2^{-31}. *If an attacker has the ability to control the inputs of the device (hence the fault-free outputs) and can inject arbitrary error patterns at the output, let* (v, y) *be an input-output pair, i.e. y is the output of the device when the input to the device is v. Then the attacker can easily derive an error pattern* $e^* = (e_y^*, e_f^*)$, $e_y^*, e_f^* \in GF(2^{32}), e_y^* \neq \mathbf{0}$ *that will be masked by y, i.e.* $(y \oplus e_y^*)^3 \oplus y^3 \oplus e_y^* = \mathbf{0}$. *During the attack, the attacker can simply input v to the device and inject the corresponding* $e^* = (e_y^*, e_f^*)$ *at the output of the device. In this case, the attack will always be successful.*

The above example assumes an advanced attacker model, where the attacker knows every detail of the cryptographic device including the error detecting code used to protect the device. The attacker can select specific inputs to the device during fault injection attacks. Moreover, the attacker is also able to inject any specific error pattern at the output of the device. In this case, the attacker has full control of not only the nonzero error $e = (e_y, e_f)$, but also the fault-free output y and the faulty output $\tilde{y} = y \oplus e_y$. Under this attacker model, all previous protection architectures based on error detecting codes will not be sufficient. An architecture that can still provide a guaranteed fault detection probability under the above attacker model is called **strongly secure cryptographic architecture**. Correspondingly, a coding technique that can be used to build strongly secure cryptographic devices is called **algebraic manipulation detection code** (AMD) (Dodis et al., 2006).

The constructions of AMD codes presented in this Chapter are based on introducing randomness into the information bits of the code. We describe the architecture of strongly secure cryptographic devices protected by these codes. In the described architecture, the redundant bits of the code are determined not only by the output y of the original device but also by the random data x generated by a true random number generator, which is incorporated into most cryptographic devices by default for key initialization, random pad computation, challenge generation, etc (Sunar et al., 2007b). We assume that both the original cryptographic devices and the true random number generator may be attacked. The attacker is able to distort x by injecting a specific additive error e_x ($\tilde{x} = x \oplus e_x$). We will show that under the most advanced attacker model described in this Section, the cryptographic devices protected by the presented AMD codes can still have a high error (fault) detecting probability.

6.2 Definitions and Bounds for Algebraic Manipulation Detection Codes

Throughout the Chapter we denote by $\oplus$ the addition in $GF(q), q = 2^r$. All the results presented in the Chapter can be easily generalized to the case where $q = p^r$ (p is a prime).

A 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. We will assume that y is a k-bit information, x is an m-bit uniformly distributed random vector (generated by a random number generator) and $f(y,x)$ is an r-bit redundant portion of the message $(y, x, f(y,x))$.

Definition 6.2.1 ***(Security Kernel)*** *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 exits y such that $f(y \oplus e_y, x \oplus e_x) \oplus f(y,x) = e_f$*

is satisfied for all x.

$$K_S = \{e | \exists y, f(y \oplus e_y, x \oplus e_x) \oplus f(y,x) = e_f, \forall x\}. \tag{6.1}$$

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. For the case of communication channels we assume that an attacker can select any k-bit vector y as the information bits of a message $(y, x, f(y,x))$ and any error $e = (e_y, e_x, e_f)$ that distorts the message. For the case of computation channels, we assume the attacker can inject faults that manifest as $e \in K_S$ at the output of the device and select y for which e is always masked. Under the above attacker model for communication or computation channels, the attacker can always mount a successful attack. Thereby an AMD code that can provide a guaranteed error detecting probability under the above strong attacker model should have no errors in the security kernel except for the all zero vector in $GF(2^n)$, where $n = k + m + r$ is the length of the code.

Definition 6.2.2 *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 $GF(2^n)$, $n = k + m + r$.*

AMD codes $V = \{(y, x, f(y,x))\}$ have no undetectable errors no matter how the attacker select $e = (e_y, e_x, e_f)$ and y. AMD codes for the case $m = r$ and $k = br$ were introduced in (Dodis et al., 2006) and were used in (Cramer et al., 2008) for robust secret sharing schemes and for robust fuzzy extractors.

For the (k, m, r) AMD code V, denote by $Q_V(y, e)$ the probability of missing an error e once y is fixed. Then $Q_V(y, e)$ can be computed as the fraction of random vectors x such that e is masked.

$$Q_V(y,e) = 2^{-m} |\{x \,|\, (y, x, f(y,x)) \in V, (y \oplus e_y, x \oplus e_x, f(y,x) \oplus e_f) \in V\}|. \tag{6.2}$$

For a (k, m, r) AMD code $V = \{(y, x, f(y,x)),\ y \in GF(2^k),\ x \in GF(2^m),\ f(y,x) \in GF(2^r)\}$, for any given $y^* \in GF(2^k)$ and $e^* = (e_y^*, e_x^*, e_f^*), e_y^* \in GF(2^k), e_x^* \in GF(2^m), e_f^* \in GF(2^r)$, $f(y^* \oplus e_y^*, x \oplus e_x^*) \oplus e_f^*$ considered as functions of $x \in GF(2^m)$ should all be different.

Example 6.2.1 *Let $k = m = tr$, $y = (y_0, y_1, \cdots, y_{t-1}), y_i \in GF(2^r)$ be the information digits and $x = (x_0, x_1, \cdots, x_{t-1}), x_i \in GF(2^r)$ be the random digits. Let $f(y,x) = x_0 \cdot y_0 \oplus x_1 \cdot y_1 \oplus \cdots \oplus x_{t-1} \cdot y_{t-1}$ be the encoding function, where all the operations are in $GF(2^r)$.*

It is easy to verify that when $e_y = 0$, for any e_x and e_f (e_x, e_f are not both 0), there always exist y such that $e = (0, e_x, e_f), e \neq 0$ will not be detected for all x. Thereby, this code is not a AMD code. In this case, K_S contains all vectors $e = (0, e_x, e_f)$.

Suppose $e_y = (e_{y_0}, e_{y_1}, \cdots, e_{y_{t-1}}), e_{y_i} \in GF(2^r)$ is always non-zero. Without lost of generality, let us assume $e_{y_0} \neq 0$. Then the monomial $e_{y_0} \cdot x_0$ will appear in the error masking equation $f(x \oplus e_x, y \oplus e_y) \oplus f(y,x) \oplus e_f = 0$. Since $e_{y_0} \neq 0$, for every e, y and $x_0, x_1, \cdots, x_{t-1}$, there is an unique solution for x_0. Thereby the error is masked with probability 2^{-r}. In this case, $K_S = \{0\}$ if $e_y \neq 0$.

Let C be a q-ary code ($q = 2^r$) of length 2^m. Codewords of C can be represented as $(f(0), f(\gamma^0), f(\gamma^1), \cdots, f(\gamma^{2^m-2}))$, where γ is a primitive element of $GF(2^m)$. (In the rest of the Chapter, we say $f(x) \in C$ if $(f(0), f(\gamma^0), f(\gamma^1), \cdots, f(\gamma^{2^m-2})) \in C$.) Let us define the orbit of $f(x)$ as

$$Orb(f) = \{\varphi | \varphi(x) = f(x \oplus e_x) \oplus e_f, e_x \in GF(2^m), e_f \in GF(2^r)\}. \qquad (6.3)$$

(We assume that any two elements $\varphi_i, \varphi_j \in Orb(f), i \neq j$ are different, i.e. $\varphi_i(x)$ and $\varphi_j(x)$ are not the same function.)

We note that for any $f(x) \in C$, $1 \leq |Orb(f)| \leq q2^m = 2^{m+r}$. If $|Orb(f)| = 2^{m+r}$, then for any e_x and e_f there exists x such that $f(x) \neq f(x \oplus e_x) \oplus e_f$. Moreover, if $\varphi(x) \notin Orb(f)$, then $Orb(\varphi) \bigcap Orb(f) = \emptyset$.

Definition 6.2.3 *We will say that a q-ary ($q = 2^r$) code C of length 2^m is a **code with full orbit** if for any $f(x) \in C$, $|Orb(f)| = 2^{m+r}$ and $Orb(f) \subseteq C$.*

Any q-ary code C of length 2^m with full orbit is a union of disjoint orbits of size $q2^m$. The size of C is a multiple of $q2^m$. We note that codes with full orbit are nonlinear and for any code C with full orbit, $\mathbf{0} \in GF(2^m)$ is not a codeword of C.

Example 6.2.2 *Let C be a binary code of length 8 and Hamming distance 2 containing all vectors with an odd number of 1's. Let $y = (y_0, y_1, y_2), y_i \in GF(2)$ and $f_y(x) = y_0 \cdot x_0 \oplus y_1 \cdot x_1 \oplus y_2 \cdot x_2 \oplus x_0 \cdot x_1 \cdot x_2$. It is easy to verify that for any $y \in GF(2^3)$, $|Orb(f_y)| = 16$. All the codewords in C can be represented as $(\varphi(0), \varphi(\gamma^0), \cdots, \varphi(\gamma^6))$, where $\varphi \in Orb(f_y)$ for some $y \in GF(2^3)$ and γ is a primitive element of $GF(2^3)$. Thus C is a code with full orbit and $|C| = |\cup_{y \in GF(2^3)} Orb(f_y)| = 128$.*

The optimal AMD code should minimize $\max_{y,e \neq 0} Q_V(y,e)$ among all codes with the same parameters. Thus, the criterion we use to construct good AMD codes is

$$\min_{V \in V_{k,m,r}} \max_{y,e \neq 0} Q_V(y,e), \tag{6.4}$$

where $V_{k,m,r}$ is the set of all (k,m,r) error detecting codes.

Let $Q_V = \max_{y,e \neq 0} Q_V(y,e)$ and $Q(k,m,r) = \min_{V \in V_{k,m,r}} Q_V$. Denote by $\hat{d}_q(2^m, M)$ the maximum Hamming distance of a q-ary ($q = 2^r$) code of length 2^m with full orbit containing M codewords. Obviously,

$$\hat{d}_q(2^m, M) \leq d_q(2^m, M), \tag{6.5}$$

where $d_q(2^m, M)$ is the maximum possible Hamming distance of a q-ary code with length 2^m and M codewords.

We next present a lower bound for $Q(k,m,r)$. The constructions of codes providing tight upper bounds for $Q(k,m,r)$ can be found in Section 6.3.

Theorem 6.2.1 *For any (k,m,r) AMD code, where k is the number of information bits, m is the number of random bits and r is the number of redundant bits,*

$$\begin{aligned} Q(k,m,r) &= \min_{V \in V_{k,m,r}} \max_{y,e \neq 0} Q_V(y,e) \\ &\geq 1 - 2^{-m} d_q(2^m, M), \end{aligned} \tag{6.6}$$

where $d_q(2^m, M)$ is the maximum possible Hamming distance of a (not necessarily systematic) q-ary code C ($q = 2^r$) with length 2^m and $M = |C| = 2^{k+m+r}$ codewords.

Proof Let V be a (k, m, r) AMD code composed of vectors $(y, x, f(y, x))$, where $y \in GF(2^k), x \in GF(2^m)$ and $f(y, x) \in GF(2^r)$. When y is fixed, $f(y, x)$ is a function of x. Let us denote this function by $f_y(x)$. Since V is an AMD code, $f_y(x \oplus e_x) \oplus e_f$ is not the same as $f_{y'}(x \oplus e'_x) \oplus e'_f$ for any $y, y', e_x, e'_x, e_f, e'_f$, assuming that elements of at least one of the pairs (y, y'), (e_x, e'_x) and (e_f, e'_f) are not equal. Thereby, for different y, e_x and e_f, $f_y(x \oplus e_x) \oplus e_f$ corresponds to 2^{k+m+r} different functions.

Let $C_V = \cup_{y \in GF(2^k)} Orb(f_y)$ be a q-ary ($q = 2^r$) code of length 2^m with full orbit. Then $|Orb(f_y)| = 2^{m+r}$, $|C| = 2^{k+m+r}$ and $Q_V = max_{y,e \neq 0} Q(y, e) = 1 - 2^{-m} d(C_V)$, where $d(C_V)$ is the Hamming distance of C_V. By (6.5) and (6.6) we have

$$\begin{aligned} Q(k, m, r) &= 1 - 2^{-m} \max_{V \in V_{k,m,r}} d(C_V) \\ &\geq 1 - 2^{-m} \hat{d}_q(2^m, M) \\ &\geq 1 - 2^{-m} d_q(2^m, M). \blacksquare \end{aligned} \tag{6.7}$$

Theorem 6.2.1 shows the relationship between the worst case error masking probability Q_V for an AMD code V and the Hamming distance of the corresponding code C_V with full orbit. The exact value of $\hat{d}_q(2^m, M)$ is hard to derive. However, the Hamming distance of C_V should not exceed the maximum possible distance for a q-ary code with length 2^m and 2^{k+m+r} codewords, $q = 2^r$. $d_q(2^m, M)$ can be estimated by classical bounds from coding theory such as the Hamming bound, the Johnson bound, the Singleton bound, the Plotkin bound, etc (MacWilliams and Sloane, 1998).

When $d_q(2^m, M)$ is estimated by the Singleton bound, $Q(k, m, r)$ can be written in a compact form as it is shown in the following Corollary.

Corollary 6.2.1 *For any (k, m, r) AMD code,*

$$Q(k, m, r) \geq \lceil \frac{k+m}{r} \rceil 2^{-m}. \tag{6.8}$$

Proof According to the Singleton bound, for any q-ary code with length n and distance d, $|C_V| \leq q^{n-d+1}$. For the code C_V in the proof of Theorem 6.2.1, $n = 2^m$, $q = 2^r$ and $|C_V| = 2^{k+m+r}$. Therefore $2^{k+m+r} \leq 2^{r(2^m-d+1)}$, or equivalently $d \leq 2^m - \lceil \frac{k+m}{r} \rceil$. Then from (6.6), we have (6.8). ■

Optimal (k, m, r) AMD codes attain the equality in (6.6) and minimize the worst case error masking probability among all codes with the same parameters.

Definition 6.2.4 *A (k, m, r) AMD code V is **optimal** iff*

$$\max_{y, e \neq 0} Q_V(y, e) = 1 - 2^{-m} d_q(2^m, M), q = 2^r, M = 2^{k+m+r}.$$

Example 6.2.3 *Let $k = m = 3$ and $r = 1$. According to (6.8), $Q(3, 3, 1) \geq \frac{6}{8}$. Let V be the code composed of all vectors $(y, x, f(y, x))$, where $y, x \in GF(2^3)$ and*

$$f(y, x) = x_0 \cdot x_1 \cdot x_2 \oplus x_0 \cdot y_0 \oplus x_1 \cdot y_1 \oplus x_2 \cdot y_2, f(y, x) \in GF(2). \tag{6.9}$$

The error masking equation is $f(x \oplus e_x, y \oplus e_y) \oplus f(y, x) = e_f$, which is a polynomial of x with degree 2. The function on the left hand side of the error masking equation corresponds to a codeword of the second order binary Reed-Muller code $RM_2(2, 3)$ with 3 variables (MacWilliams and Sloane, 1998). Any codeword of $RM_2(2, 3)$ has a Hamming weight of at least 2. Thus the number of solutions for the error masking equation is upper bounded by 6. V is a AMD code with $Q_V = \frac{6}{8}$. It follows from (6.8) that this code is optimal and $Q(3, 3, 1) = 0.75$.

Remark 6.2.1 *We note that AMD codes V with Q_V close to 1 may still be very useful for channels with memories where errors tend to repeat themselves, e.g. for the protection of cryptographic hardware against fault injection attacks when errors have a high probability to repeat for several clock cycles (slow fault injection attacks and lazy channels). To our best knowledge, this assumption can be true for most of*

the modern fault injection mechanisms due to their limited timing resolutions (Skorobogatov and Anderson, 2003; Schmidt and Hutter, 2007; Skorobogatov, 2010). In this case a repeating error will be ultimately detected by AMD codes after it distorts several consecutive messages.

In the next section, we will present several general constructions of AMD codes. Some of the generated codes are optimal with respect to the lower bounds (6.6) or (6.8).

6.3 Constructions of AMD Codes

The codewords of a (k, m, r) AMD code V are in the format $(y, x, f(y, x))$, where $y \in GF(2^k), x \in GF(2^m)$ and $f(y, x) \in GF(2^r)$. When y is fixed, $f_y(x)$ is a function of x. In the proof of Theorem 6.2.1, we have shown that the necessary condition for V to be an AMD code is that $f_y(x \oplus e_x) \oplus e_f$ cannot be the same function as $f_{y'}(x \oplus e'_x) \oplus e'_f$ for any $y, y', e_x, e'_x, e_f, e'_f$, assuming elements in at least one of the pairs (y, y'), (e_x, e'_x) and (e_f, e'_f) are not equal.

To compute $Q_V(y, e)$ for the code V, the error masking equation $f(x \oplus e_x, y \oplus e_y) \oplus f(y, x) \oplus e_f = 0$ should be evaluated for all 2^m possible $x \in GF(2^m)$.

We will say that an AMD code $V = \{(y, x, f(y, x))\}$ is based on code C_V if the error masking polynomial $f(y \oplus e_y, x \oplus e_x) \oplus f(y, x) \oplus e_f$ is a codeword of C_V for all y, e_x, e_y and e_f. Let us re-write $f(y, x)$ as $f(y, x) = A(x) \oplus B(y, x)$, where $A(x)$ is independent of y. We next show that by selecting $A(x)$ and $B(y, x)$ based on different error detecting codes such as the Generalized Reed-Muller codes and the Reed-Solomon codes, we can construct good (and in many cases optimal) AMD codes for different k and different $Q_V = \max_{y,e\neq 0} Q_V(y, e)$ for given m and r.

6.3.1 Constructions Based on the Generalized Reed-Muller Codes

Let $x = (x_0, x_1, \cdots, x_{t-1}), x_i \in GF(q)$, $q = 2^r$. A b^{th} order q-ary Generalized Reed-Muller code $GRM_q(b,t)$ (Assmus et al., 1995) with t variables $(1 \leq b \leq t(q-1))$ consists of all codewords $(f(0), f(\gamma^0), \cdots, f(\gamma^{q^t-2}))$, where $f(x)$ is a polynomial of t variables $x_0, x_1, \cdots x_{t-1}$ and γ is a primitive element of $GF(q^t)$. The degree of $f(x)$ is less or equal to b.

As it is shown in (Assmus et al., 1995), the dimension of $GRM_q(b,t)$ is

$$k_{GRM_q(b,t)} = \sum_{j=0}^{t} (-1)^j \binom{t}{j} \binom{t+b-jq}{b-jq}, q = 2^r. \tag{6.10}$$

If $b = u(q-1)+v, 0 \leq v \leq q-2$. Then the distance of $GRM_q(b,t)$ is $d_{GRM_q(b,t)} = (q-v)q^{t-u-1}$ (Assmus et al., 1995). Suppose $b+2 = \alpha(q-1)+\beta \leq t(q-1), 0 \leq \alpha \leq t, 0 \leq \beta \leq q-2$.

Let

$$A(x) = \begin{cases} \bigoplus_{i=0}^{t-1} x_i^{b+2} & \text{if } \alpha = 0, b \text{ is odd;} \\ \bigoplus_{i=1}^{t-1} x_0 x_i^{b+1}, t > 1 & \text{if } \alpha = 0, b \text{ is even;} \\ \bigoplus_{i=0}^{t-1} x_i^{\beta} \prod_{j=1}^{\alpha} x_{|i+j|_t}^{q-1} & \text{if } \alpha \neq 0, \alpha \neq t \\ \prod_{i=0}^{\alpha-1} x_i^{q-1} & \text{if } \alpha = t \end{cases} \tag{6.11}$$

where $x_i \in GF(2^r)$, $|i+j|_t$ is the modulo t addition, $\bigoplus$ is the sum and $\prod$ is the product in $GF(2^r)$.

Let

$$B(y,x) = \bigoplus_{1 \leq j_0+j_1+\cdots+j_{t-1} \leq b+1} y_{j_0,j_1,\cdots,j_{t-1}} \prod_{i=0}^{t-1} x_i^{j_i}, \tag{6.12}$$

where $y_{j_0,j_1,\cdots,j_{t-1}} \in GF(2^r), x_i \in GF(2^r)$, $\prod_{i=0}^{t-1} x_i^{j_i}$ is a monomial of $x_0, x_1, \cdots, x_{t-1}$

of a degree between 1 and $b+1$ and $\prod_{i=0}^{t-1} x_i^{j_i} \notin \Delta B(y,x)$, where

$$\Delta B(y,x) = \begin{cases} \{x_0^{b+1}, x_1^{b+1}, \cdots, x_{t-1}^{b+1}\} & \text{if } \alpha = 0, b \text{ is odd;} \\ \{x_1^{b+1}, x_0x_1^b, x_0x_2^b, \cdots, x_0x_{t-1}^b, t>1\} & \text{if } \alpha = 0, b \text{ is even;} \\ \{x_i^\beta x_{|i+1|_t}^{q-2} \prod_{j=2}^{\alpha} x_{|i+j|_t}^{q-1}, 0 \le i \le t-1\} & \text{if } \alpha \neq 0. \end{cases} \quad (6.13)$$

It follows from (6.13) that when $\alpha = t$, $\Delta B(y,x) = \{x_i^{q-2} \prod_{j \neq i} x_j^{q-1}, 0 \le i \le t-1\}$.

Example 6.3.1 *Let $r = 3, q = 8, t = 2$ and $b = 10$. Since $b+2 = 12 = \alpha(q-1)+\beta$, we have $\alpha = 1$ and $\beta = 5$. By (6.11) and (6.13), we have $A(x) = x_0^5x_1^7 \oplus x_0^7x_1^5$ and $\Delta B(y,x) = \{x_0^5x_1^6, x_0^6x_1^5\}$. It is easy to verify that $A(x \oplus e_x) \oplus A(x) \oplus B(y \oplus e_y, x \oplus e_x) \oplus B(y,x)$ is always a non-zero polynomial corresponding to a codeword in $GRM_8(11,2)$.*

AMD codes can be constructed based on $A(x), B(y,x)$ and the Generalized Reed-Muller codes as shown in the next Theorem.

Theorem 6.3.1 *Let $f(y,x) = A(x) \oplus B(y,x)$ be a q-ary polynomial with $y_{j_0,j_1,\cdots,j_{t-1}} \in GF(q)$ as coefficients and $x \in GF(q^t)$ as variables, where $1 \le b \le t(q-1)-2, q = 2^r$ and $A(x), B(y,x)$ are as shown above. Suppose $b+2 = \alpha(q-1)+\beta$ and $b+1 = u(q-1)+v$, $0 \le \alpha, u \le t, 0 \le \beta, v \le q-2$. Assume $b+2 \neq t(q-1)-1$. Then the code V composed of all vectors $(y,x,f(y,x))$ is an AMD code with $m = tr$,*

$$\begin{aligned} k &= (k_{GRM_q(b+1,t)} - t - 1)r \\ &= (\sum_{i=0}^{t} (-1)^i \binom{t}{i} \binom{t+b+1-iq}{b+1-iq} - 1 - t)r, \end{aligned} \quad (6.14)$$

and

$$\begin{aligned} Q_V &= 1 - d_{GRM_q(b+1,t)} 2^{-m} \\ &= 1 - (2^r - v)2^{-(u+1)r}. \end{aligned} \quad (6.15)$$

Thus

$$Q((\sum_{i=0}^{t} (-1)^i \binom{t}{i} \binom{t+b+1-iq}{b+1-iq} - 1 - t)r, tr, r) \le 1 - (2^r - v)2^{-(u+1)r}. \quad (6.16)$$

Proof An error e is masked by V if and only if $f(y \oplus e_y, x \oplus e_x) \oplus f(y,x) \oplus e_f = 0$ for all x, which can be re-written as

$$A(x \oplus e_x) \oplus A(x) \oplus B(y \oplus e_y, x \oplus e_x) \oplus B(y,x) \oplus e_f = 0. \tag{6.17}$$

1. If $e_x = 0$ and $e_y = 0$, the error is always detected unless e_f is also 0. If $e_x = 0$ and $e_y \neq 0$, the left hand side of (6.17) is a polynomial of degree from 1 to $b+1$, which corresponds to a codeword of a $(b+1)^{th}$ order q-ary Generalized Reed-Muller code. Since $d_{GRM_q(b+1,t)} = (q-v)q^{t-u-1}$, there are at most $q^t - (q-v)q^{t-u-1}$ solutions for the error masking equation.

2. If $e_x \neq 0$, the left hand side of (6.17) does not contain any monomials of degree $b+2$ due to the fact that $A(x)$ and $A(x \oplus e_x)$ have exactly the same monomials of degree $b+2$. Moreover,

   (a) If $\alpha = 0$ and b is odd, x_i^{b+1} appears in (6.17) iff x_i is distorted, $0 \leq i \leq t-1$;

   (b) If $\alpha = 0$ and b is even, x_1^{b+1} appears in (6.17) iff x_0 is distorted, $x_0 x_i^b$ appears in (6.17) iff x_i is distorted $1 \leq i \leq t-1$;

   (c) If $\alpha \neq 0$, since $b+2 \neq t(q-1)-1$, $x_i^{\beta} x_{|i+1|_t}^{q-2} \prod_{j=2}^{\alpha} x_{|i+j|_t}^{q-1}$ appears in (6.17) iff $x_{|i+1|_t}$ is distorted, $0 \leq i \leq t-1$. (When $\alpha = t$, it is equal to say that ${x_i}^{q-2} \prod_{j \neq i} x_j^{q-1}$ appears in (6.17) if x_i is distorted.)

   Thereby, (6.17) always contains monomials of degree $b+1$, the left hand side of the error masking equation again is a codeword in $GRM_q(b+1,t)$. Thus the number of solutions for the error masking equation is still upper bounded by $q^t - (q-v)q^{t-u-1}$.

Thus for any fixed y and e, the probability Q_V of error masking is upper bounded by

$$(q^t - (q-v)q^{t-u-1})q^{-t} = 1 - (2^r - v)2^{-(u+1)r}.$$

The left hand side of (6.17) contains monomials of a degree from 1 to $b+1$ except for the t monomials from $\Delta B(y,x)$. Hence the number of different monomials in $B(y,x)$ is

$$k_{GRM_q(b+1,t)} - 1 - t = \sum_{i=0}^{t}(-1)^i \binom{t}{i}\binom{t+b+1-iq}{b+1-iq} - 1 - t. \tag{6.18}$$

The number, k, of bits in y is equal to the number of monomials in $B(y,x)$ multiplied by r, which is

$$(\sum_{i=0}^{t}(-1)^i \binom{t}{i}\binom{t+b+1-iq}{b+1-iq} - 1 - t)r.\blacksquare \tag{6.19}$$

Example 4 (Continued) For the code shown in Example 6.3.1, $k = 55\times 3 = 165$. Since $b = 10 = u(q-1)+v, q = 8$, we have $u = 1$ and $v = 3$. The worst case error masking probability is $Q_V = 1-5\times 2^{-6}$. Thus by (6.8), $1-7\times 2^{-6} \le Q_V(165,6,3) \le 1-5\times 2^{-6}$.

Corollary 6.3.1 *When $b = t(q-1)-2, q = 2^r$, codes generated by Theorem 6.3.1 are optimal. We have*

$$Q(2^{tr}r - tr - 2r, tr, r) = 1 - 2^{-tr+1}. \tag{6.20}$$

Proof According to (6.8), $Q(2^m r - m - 2r, m, r) \ge 1 - 2^{-m+1}$, where $m = tr$. The number, k, of information bits for the AMD code V generated by Theorem 6.3.1 is $(q^t - t - 2)r$. Thus, we have

$$b + 1 = t(q-1) - 1 = (t-1)(q-1) + q - 2.$$

Thereby $u = t-1$ and $v = q-2$. The worst case error masking probability for V is

$$Q_V = 1 - (2^r - (q-2))2^{-tr} = 1 - 2^{-tr+1}.$$

The code is optimal with respect to the lower bound (6.8). ■

Special Case: $r = 1$

For this case the dimension of a $(b+1)^{th}$ order binary Reed-Muller code of t variables is $k_{RM_2(b+1,t)} = \sum_{i=0}^{b+1} \binom{t}{i}$ $(t = m)$ (MacWilliams and Sloane, 1998). The distance of $RM_2(b+1,t)$ is $d_{RM_2(b+1,t)} = 2^{t-b-1}$. As a result, the dimension of the resulting AMD code V constructed by Theorem 6.3.1 is $k = \sum_{i=0}^{b+1} \binom{t}{i} - t - 1$. The worst case error masking probability of the code is $Q_V = 1 - 2^{-(b+1)}$.

Example 6.3.2 *Suppose* $m = 7$ *and* $r = 1$. *Let* $b = 1$ *and*

$$f(y,x) = x_0 \cdot x_1 \cdot x_2 \oplus x_3 \cdot x_4 \cdot x_5 \oplus x_0 \cdot x_3 \cdot x_6 \oplus \sum_{i=0}^{6} x_i \cdot y_i. \tag{6.21}$$

It is easy to verify that $f(y \oplus e_y, x \oplus e_x) \oplus f(y,x) \oplus e_f$ *is a polynomial of degree* 2, *which is a codeword of* $RM_2(2,7)$. *The distance of* $RM_2(2,7)$ *is* 32. *The worst case error masking probability of the resulting AMD code is* $Q_V = \frac{3}{4}$.

Corollary 6.3.2 *When* $q = 2$, *the code* V *generated by Theorem 6.3.1 is a* $(\sum_{i=0}^{b+1} \binom{t}{i} - t - 1, t, 1)$ *AMD code with* $Q_V = 1 - 2^{-(b+1)}$. *The code is optimal when* $b = t - 2$.

Proof Corollary 6.3.2 follows from Corollary 6.3.1 with $r = 1$. ■

Special Case: $b \leq q - 3$

Another special case of Theorem 6.3.1 is the case $b \leq q-3$. In this case $k_{GRM_q(b+1,t)} = \binom{t+b+1}{t}$ and $d_{GRM_q(b+1,t)} = (q-b-1)q^{t-1}$ (Assmus et al., 1995). The dimension of the resulting AMD code is $(\binom{t+b+1}{t} - 1 - t)r$. The worst case error masking probability is $(b+1)2^{-r}$.

Corollary 6.3.3 *When* $b \leq q - 3$, *the code* V *generated by Theorem 6.3.1 is a* $((\binom{t+b+1}{t} - 1 - t)r, tr, r)$ *AMD code with* $Q_V = (b+1)2^{-r}$.

Proof By Theorem 6.3.1, k and Q_V of the AMD code V can be easily derived from the parameters of $GRM_q(b+1,t)$, $b \leq q-3$, $q = 2^r$. ■

Example 6.3.3 *When $b+1 = t(q-1)-3$, the dimension of the AMD code generated by Theorem 6.3.1 is $2^{tr} - 1 - t - \binom{t+2}{t}$. The worst case error masking probability for the code is $Q_V = 1 - 4 \times 2^{-tr}$. According to (6.8),*

$$Q((q^t - 1 - t - \binom{t+2}{t})r, tr, r) \geq (2^{tr} - 1 - \binom{t+2}{t})2^{-tr}. \tag{6.22}$$

Thereby we have

$$1 - (1 + \binom{t+2}{t})2^{-tr} \leq Q((2^{tr} - 1 - t - \binom{t+2}{t})r, tr, r) \leq 1 - 4 \cdot 2^{-2r}. \tag{6.23}$$

For example, for $t = 2$,

$$1 - 7 \cdot 2^{-2r} \leq Q((2^{2r} - 9)r, 2r, r) \leq 1 - 4 \cdot 2^{-2r}. \tag{6.24}$$

When $b = 1$ $B(y, x)$ is the quadratic form $x_0 \cdot y_0 \oplus x_1 \cdot y_1 \oplus \cdots \oplus x_{t-1} \cdot y_{t-1}$, where all the operations are in $GF(2^r)$. If $e_y \neq 0$, it is easy to verify that the number of solutions for (6.17) is upper bounded by q^{t-1}.

Special Case: $t = 1$ (Dodis et al., 2006; Cramer et al., 2008)

When $t = 1$ and b is odd, $A(x) = x^{b+2}$ and $B(y, x) = x \cdot y_0 \oplus x^2 \cdot y_1 \oplus \cdots \oplus x^b \cdot y_{b-1}$. The code generated by Theorem 6.3.1 coincides with the construction shown in (Dodis et al., 2006; Cramer et al., 2008).

Corollary 6.3.4 *(Dodis et al., 2006; Cramer et al., 2008) When $b \leq q-3$ is an odd number, the code V composed of all vectors $(y, x, f(y, x))$, where $y \in GF(q^{bt}), x \in GF(q), q = 2^r$ and $f(y, x) = x^{b+2} \oplus x \cdot y_0 \oplus x^2 \cdot y_1 \oplus \cdots x^b \cdot y_{b-1}, f(y, x) \in GF(q)$, is an optimal (br, r, r) AMD code with $Q_V = \max_{y, e \neq 0} Q_V(y, e) = (b+1)2^{-r}$. Thereby, $Q(br, r, r) = (b+1)2^{-r}$.*

Proof For codes generated by Corollary 6.3.4, $m = r$, $k = br$ and $Q_V = (b+1)2^{-r}$. According to Corollary 6.2.1, $Q(k, m, r) \geq \lceil (k+m)r^{-1} \rceil 2^{-m}$. Thereby we have $Q(br, r, r) = (br + r)r^{-1}2^{-m} = (b+1)2^{-r}$. ■

Remark 6.3.1 *One limitation of Corollary 6.3.4 is that b can only be an odd number when the characteristic of the field $GF(q)$ is 2. Otherwise, $A(x \oplus e_x)$ for $A(x) = x^{b+2}$ and $e_x \neq 0$ does not contain any monomial of degree $b+1$ since $(b+1)x^{b+1} = 0$. The resulting code is not a secure AMD code as pointed out in (Cramer et al., 2008). When b is even, $A(x)$ can be chosen as x^{b+3}. In this case, $Q_V = (b+2)2^{-r}$.*

Remark 6.3.2 *When $t = 1$, the left hand side of the error masking equation $f(y \oplus e_y, x \oplus e_x) \oplus f(y, x) \oplus e_f = 0$ is a codeword of an extended q-ary Reed-Solomon code, $q = 2^r$ (MacWilliams and Sloane, 1998).*

When $t > 1$, codes V generated by Theorem 6.3.1 may have larger number of codewords than codes generated by Corollary 6.3.4 ($t = 1$), assuming the two codes have the same Q_V and the same r.

Example 6.3.4 *Suppose $r = 16$, $Q_V = 2^{-14}$. Then for $t = 1$ and $b = 3$, for codes generated by Corollary 6.3.4, the maximum number of codewords is $2^{br} = 2^{48}$. When $t > 1$, the maximum number of codewords for codes generated by Theorem 6.3.1 depends not only on b but also on t. When $t = 2$, for example, the number of codewords of codes generated by Theorem 6.3.1 can be $2^{(\binom{t+b+1}{t}-1-t)r} = 2^{192}$.*

For AMD codes generated by Theorem 6.3.1, k and m are both multiples of r. We will now present three modification methods, which can largely increase the flexibility of parameters of the resulting AMD codes.

Theorem 6.3.2 *Suppose there exists an (k, m, r) AMD code generated by Theorem 6.3.1 with $r \geq 1$, $m = tr$, $k = sr$ and $Q_V = \max_{y,e\neq 0} Q_V(y, e)$.*

1. *For the same r, m and $r \leq k < sr$, a shortened AMD code with the same Q_V can be constructed by appending $0's$ to y so that $(\mathbf{0}, y) \in GF(2^{sr})$ and then apply the same encoding procedure as for the (sr, tr, r) code.*

2. *For the same m, k and $1 \leq r' < r$, an AMD code can be constructed by deleting $r - r'$ redundant bits from each codeword of the original (k, m, r) code. The maximum error masking probability of the resulting code will be $\min\{Q_V 2^{r-r'}, 1\}$.*

3. *Suppose there exists a* (k_1, m, r_1) *AMD code* V_1 *with* $\max_{y,e\neq 0} Q_{V_1}(y,e) = Q_{V_1}$ *and another* (k_2, m, r_2) *AMD code* V_2 *with* $\max_{y,e\neq 0} Q_{V_2}(y,e) = Q_{V_2}$. *By computing the redundant bits of the two codes separately and then concatenating them, we can construct a* $(k_1 + k_2, m, r_1 + r_2)$ *AMD code with* $Q_V \leq \max\{Q_{V_1}, Q_{V_2}\}$.

Proof 1. For codes constructed by Theorem 6.3.1, the error masking equation $f(y \oplus e_y, x \oplus e_x) \oplus f(y,x) \oplus e_f = 0$ is a polynomial of degree at most $b+1$, where $x_0, x_1, \cdots, x_{t-1}$ are variables and y, e_y, e_x and e_f are coefficients. Obviously, modifications of coefficients do not change the maximum possible degree of the polynomial thus do not change the maximum number of solutions for the error masking equation.

2. For the (sr, tr, r) AMD code with $Q_V = \max_{y,e} Q_V(y,e)$, every fixed e_y, e_x, e_f and y is masked by no more than $Q_V 2^m$ different x. After deleting $r - r'$ bits from the values of the function $f(y,x)$, the vectors (y,x) which previously mapped to $f(y,x)$ that are different in the deleted $r - r'$ bits will now map to the same value of the redundant bits. Thereby, when $r - r'$ bits are deleted, for any fixed e_y, e_x, e_f and y, the error is masked by at most $\min\{Q_V 2^{m+r-r'}, 2^m\}$ different x.

3. For the concatenated $(k_1 + k_2, m, r_1 + r_2)$ AMD code V, codewords are

$$(y_1, y_2, x, f_1(y_1, x), f_2(y_2, x)),$$

where $y_1 \in GF(2^{k_1}), y_2 \in GF(2^{k_2}), x \in GF(2^m)$, $f_1(y_1, x) \in GF(2^{r_1})$ and $f_2(y_2, x) \in GF(2^{r_2})$. For any $y = (y_1, y_2)$ and any error $e = (e_{y_1}.e_{y_2}, e_x, e_{f_1}, e_{f_2})$, $e_{y_1} \in GF(2^{k_1})$, $e_{y_2} \in GF(2^{k_2})$, $e_x \in GF(2^m)$, $e_{f_1} \in GF(2^{r_1})$, $e_{f_2} \in GF(2^{r_2})$, denote $N(y,e)$ a number of $x's$ satisfying simultaneously the following two error

masking equations

$$\begin{cases} f_1(y_1 \oplus e_{y_1}, x \oplus e_x) \oplus f_1(y_1, x) \oplus e_{f_1} = 0 \\ f_2(y_2 \oplus e_{y_2}, x \oplus e_x) \oplus f_2(y_2, x) \oplus e_{f_2} = 0 \end{cases} \tag{6.25}$$

Suppose $Q_{V_1} \geq Q_{V_2}$, then by the definition of the error masking probability Q_V, when $e_{y_2} = e_x = e_{f_2} = 0$, we have $\max_{y,e\neq 0} N(y,e) \leq 2^m Q_{V_1} = 2^m \max\{Q_{V_1}, Q_{V_2}\}$. Thereby, $Q_V \leq 2^{-m} \max_{y,e\neq 0} N(y,e) \leq \max\{Q_{V_1}, Q_{V_2}\}$ ■

Concatenation of L copies of a (k,m,r) AMD code constructed by Theorem 6.3.1 generates a (k',m',r') code with $k' = Lk, m' = m$ and $r' = Lr$. According to the Singleton bound,

$$Q(Lk,m,Lr) \geq \lceil \frac{Lk+m}{Lr} \rceil 2^{-m}. \tag{6.26}$$

When $b \leq q-3$, from Corollary 6.3.3 we have

$$Q(Lk,m,Lr) \leq (b+1)2^{-r}.$$

Thus

$$\lceil \frac{Lk+m}{Lr} \rceil 2^{-m} \leq Q(Lk,m,Lr) \leq (b+1)2^{-r}, b \leq q-3. \tag{6.27}$$

Corollary 6.3.5 *Let V be an optimal (k,m,r) AMD code with $k = sr, m \leq r$ and $Q_V = \lceil \frac{k+m}{r} \rceil 2^{-m}$. Then for any L, the (Lk,m,Lr) code V' obtained by concatenation of L copies of V is also optimal.*

Proof By part 3 of Theorem 6.3.2, we have $Q_{V'} \leq Q_V = \lceil \frac{k+m}{r} \rceil = (s+1)2^{-m}$. On another hand by (6.8), we have $Q_{V'} \geq \lceil \frac{Lk+m}{Lr} \rceil 2^{-m} = \lceil s + \frac{m}{Lr} \rceil = (s+1)2^{-m}$. ■

The concatenation of AMD codes based on $GRM_q(b+1,1)$ is optimal for $b \leq q-3$ and $Q(Lbr,r,Lr) = (b+1)2^{-r}$.

6.3.2 Constructions Based on Products of Generalized Reed-Muller Codes

Theorem 6.3.3 *Let $C_{V_i}, 1 \leq i \leq L$ be a $(b_i+1)^{th}$ order q-ary Generalized Reed-Muller code defined over t_i variables with dimension k_i and distance d_i, $q = 2^r$. Let*

V_i be an AMD code constructed based on C_{V_i} with the encoding function $f_i(y,x) = A_i(x) \oplus B_i(y,x)$ as shown in Theorem 6.3.1. Let $A(x) = \bigoplus_{i=1}^{L} A_i(x)$ and

$$B(y,x) = \bigoplus_{P_1,P_2,\cdots,P_L} y_{P_1,P_2,\cdots,P_L} \prod_{i=1}^{L} P_i, \tag{6.28}$$

where P_i is a polynomial of the t_i variables in C_{V_i}, $deg(P_i) \leq b+1, P_i \notin \Delta B_i(y,x)$ and $\prod_{i=1}^{L} P_i$ is not a constant. Then the code V defined by $f(y,x) = A(x) \oplus B(y,x)$ is a (k,m,r) AMD code V with $m = r\sum_{i=1}^{L} t_i$,

$$k = (\prod_{i=1}^{L}(k_i - t_i) - 1)r, \tag{6.29}$$

and

$$Q_V = 1 - 2^{-r\sum_{i=1}^{L} t_i} \prod_{i=1}^{L} d_i. \tag{6.30}$$

Proof The error masking polynomial $f(y \oplus e_y, x \oplus e_x) \oplus f(y,x) \oplus e_y$ is always a non-zero polynomial, which is a codeword of the product of $C_{V_i}, 1 \leq i \leq L$. The distance of the product code is $\prod_{i=1}^{L} d_i$. Hence Q_V for the AMD code is $Q_V = 1 - d2^{-m} = 1 - 2^{-r\sum_{i=1}^{L} t_i} \prod_{i=1}^{L} d_i$. By (6.13), the number of P_i such that $deg(P_i) \leq b+1$ and $P_i \notin \Delta B_i(y,x)$ is $k_i - t_i$. Thus the number of monomials in $B(y,x)$ is $\prod_{i=1}^{L}(k_i - t_i) - 1$. ($\prod_{i=1}^{L} P_i$ is not a constant.) The dimension of the AMD code V is equal to the number of monomials in $B(y,x)$ multiplied by r, which is $(\prod_{i=1}^{L}(k_i - t_i) - 1)r$. ■

In the previous Section we've seen that the error masking equation for codes generated by Corollary 6.3.4 (special case of Theorem 6.3.1 when $t = 1$) corresponds to a codeword from a q-ary extended Reed-Solomon code with length 2^m and dimension $b+2$. When $t = 1$, the AMD codes generated by Theorem 6.3.3 are based on the product of L q-ary extended Reed-Solomon codes (PRS) (Santhi, 2007). The construction and the parameters of AMD codes based on the extended PRS code are shown in the next Corollary.

Corollary 6.3.6 *When $t = 1$, AMD codes generated by Theorem 6.3.3 are based on the extended PRS codes. Suppose each extended Reed-Solomon code has dimension $b+2$ and length $q = 2^r, q \geq b+3$. Let*

$$A(x) = x_1^{b+2} \oplus x_2^{b+2} \oplus \cdots \oplus x_L^{b+2}. \tag{6.31}$$

Let

$$B(y,x) = \bigoplus_{s_1=0}^{b} \cdots \bigoplus_{s_L=0}^{b} y_{s_1,\cdots,s_L} \prod_{i=1}^{L} x_i^{s_i}, (s_1, \cdots, s_t) \neq \mathbf{0}. \tag{6.32}$$

The resulting AMD code V is a $(((b+1)^L - 1)r, Lr, r)$ code with

$$Q_V = \max_{y,e\neq 0} Q_V(y,e) = 1 - 2^{-Lr}(2^r - b - 1)^L. \tag{6.33}$$

Proof The Corollary can be easily proved by substituting the parameters of the extended Reed-Solomon codes into (6.29) and (6.30). ■

Example 6.3.5 *Let $r = 3, L = 2$ and $b = 3$. For the AMD code V generated by Corollary 6.3.6, $m = 6$ and $k = ((b+1)^L - 1)r = 45$. Each extended Reed-Solomon (RS) code has Hamming distance 5. For the extended PRS code, $d_{PRS} = 25$. Thereby the worst case error masking probability of the $(45, 6, 3)$ AMD code is $Q_V = 1 - 25 \cdot 2^{-6} = 39 \cdot 2^{-6}$.*

For codes generated by Corollary 6.3.3, the worst case error masking probability is $Q_1 = (b_1+1)2^{-r}$. For codes generated by Corollary 6.3.6, $Q_2 = 1-2^{-Lr}(2^r - b_2 - 1)^L$. Suppose the two codes have the same r and $Q_1 = Q_2$.

Let $b_1 + 1 = 2^r - u$, where $u \geq 2$. Then it can be easily proved that

$$b_2 + 1 = 2^r - 2^r(\frac{u}{2^r})^{\frac{1}{L}} = 2^r - 2^{(1-\frac{1}{L})r} u^{\frac{1}{L}}.$$

As it is illustrated by the following example, when r is large and b_1 (and b_2) is close to 2^r, the number of information bits for codes generated by Corollary 6.3.6 can be much larger than for codes generated by Corollary 6.3.3.

Example 6.3.6 *Let* $r = 8, q = 2^r = 256$, $u = 4$ *and* $m = 16$. *For codes generated by Corollary 6.3.3,* $t = 2$, $b_1 + 1 = 2^r - u = 252$, $k = (\binom{t+b_1+1}{t} - 1 - t)r = 32,131 \times 8$ *bits. For codes generated by Corollary 6.3.6,* $L = 2$, $b_2 + 1 = 2^r - 2^{(1-\frac{1}{L})r} u^{\frac{1}{L}} = 224$, $k = ((b+1)^L - 1)r = 50,175 \times 8$ *bits. These two codes have the same worst case error masking probability* Q_V. *However, the number of information bits for the AMD code based on the extended PRS code is much larger than that based on the Generalized Reed-Muller code.*

6.4 Protection of Normal Base Serial Multipliers in $GF(2^k)$

As a case study, in this section we will present architectures based on the proposed AMD codes for secure multipliers in $GF(2^k)$, which are commonly used blocks in cryptographic devices implementing the elliptic curve cryptographic algorithms (SECG, 2000), etc.

The general architecture using AMD codes to protect devices against fault injection attacks is shown in Figure 6·1. In addition to the original device, two extra blocks, the predictor and the error detecting network (EDN) are needed. The extended outputs of the fault-free device are codewords of the AMD code. As in most works discussing the protection of data-path in cryptographic devices (Karpovsky et al., 2004; Gaubatz et al., 2006), we assume that the EDN is tamper resistant and cannot be attacked by the attacker. Otherwise, an advanced attacker can easily bypass any kind of protection mechanism based on error detecting codes by forcing the error flag signal *Error* to be 0 (Figure6·1).

The hardware implementations of multipliers in $GF(2^k)$ can be categorized as parallel multipliers and serial (sequential) multipliers. Compared to parallel multipliers, serial multipliers are more area efficient and are more practical in hardware for multiplications in a large Galois field especially in small digital devices, e.g. smart phones. A digit-serial Massey-Omura multiplier can output one digit of the product

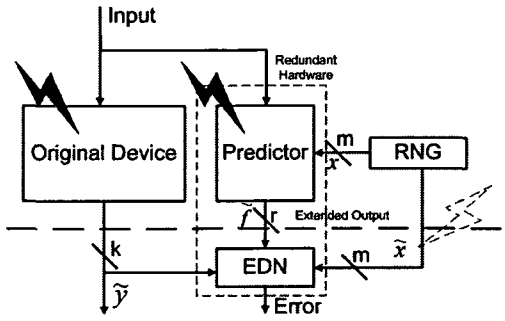


Figure 6·1: General architecture of a device protected by a (k, m, r) AMD code

per clock cycle. Suppose the length of the digit is r-bit and the output of the multiplier is k-bit. The multiplication in $GF(2^k)$ is completed in $\lceil \frac{k}{r} \rceil$ clock cycles. The digit-serial Massey-Omura multiplier (Massey and Omura, 1986) can be implemented by using r identical combinatorial blocks with cyclically shifted inputs for normal base multiplication in $GF(2^k)$.

We next estimate the area overhead for a digit-serial Massey-Omura multiplier in $GF(2^k)$ protected by AMD codes with $b = 1$ generated by Theorem 6.3.1 and Corollary 6.3.4 ($t = 1$ in Theorem 6.3.1).

When $b = 1$ in Theorem 6.3.1,

$$f(y, x) = \bigoplus_{i=0}^{t-1} x_i^3 \oplus x_i y_i = \bigoplus_{i=0}^{t-1} x_i (x_i^2 \oplus y_i).$$

If $e_y \neq 0$, $\bigoplus_{i=0}^{t-1} x_i^3$ can be omitted and $f(y, x)$ can be simplified to be $\bigoplus_{i=0}^{t-1} x_i y_i$. The structure of the predictor for AMD codes with $b = 1$ for the protection of a digit-serial multiplier in $GF(2^k)$ is shown in Figure 6·2. $x_i \in GF(2^r)$ is the random data generated by the true random number generator. y_i is the i^{th} component of the product. When k is not a multiple of r, $0's$ are appended so that all $(y_i, \mathbf{0})$ belong to $GF(2^r)$. In addition to the duplicated digit-serial multiplier in $GF(2^k)$, a r-bit register, an extra parallel multiplier and two adders in $GF(2^r)$ are also required.

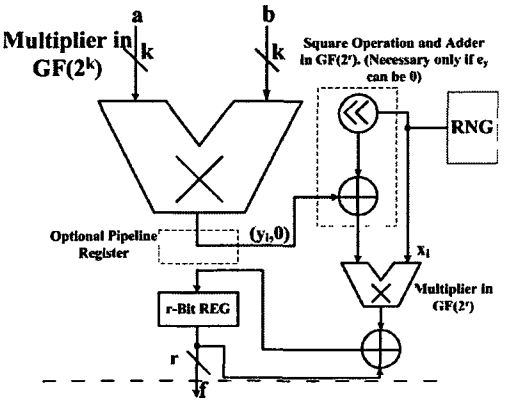


Figure 6·2: Predictor for serial Massey-Omura multiplier in $GF(2^k)$ protected by codes with $b = 1$ generated by Theorem 6.3.1

(The square operation can be implemented by cyclically shifting in normal base Galois fields.) The parallel multiplier in $GF(2^r)$ can be implemented as described in (Reyhani-Masoleh and Hasan, 2002). At every clock cycle, the digit generated by the digit-serial multiplier in $GF(2^k)$ is multiplied by a r-bit random data x_i. The result is cumulatively added and saved in the r-bit register. After t clock cycles, the redundant bits will be available in the r-bit register and will be verified by EDN to detect errors. To reduce the latency of the predictor, an optional pipeline register can be added between the original digit-serial multiplier in $GF(2^k)$ and the parallel multiplier in $GF(2^r)$ as shown by the dotted block in Figure 6·2.

When $t = 1$ and b is odd in Theorem 6.3.1,

$$\begin{aligned} f(y,x) &= x^{b+2} \oplus \bigoplus_{i=1}^{b} y_{i-1}x^i \\ &= x(y_0 \oplus x(y_1 \oplus \cdots \oplus x(y_{b-1} \oplus x^2)\cdots)) \end{aligned}$$

The structure of the predictor for the resulting AMD codes is shown in Figure 6·3. During the first clock cycle of every multiplication, the output digit y_0 ($0's$ are appended if necessary) is added to x^2 and then multiplied by x. For each of the

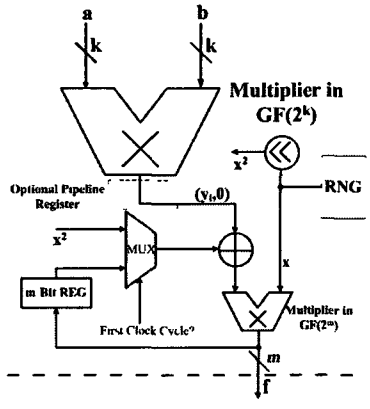


Figure 6·3: Predictor for serial Massey-Omura multiplier in $GF(2^k)$ protected by codes with $t = 1$ based on Theorem 6.3.1

Table 6.1: Hardware complexity for parallel and digit-serial Massey-Omura multipliers

Type	AND	XOR	Latency
Digital-Serial MO(Massey and Omura, 1986)	rC_N	$r(C_N - 1)$	$T_A + \lceil log_2 C_N \rceil T_X$
P. RR-MO(Reyhani-Masoleh and Hasan, 2002)	k^2	$\frac{k}{2}(C_N + k - 2)$	$T_A + \lceil log_2(C_N + 1) \rceil T_X$
Parallel RR-MO*	k^2	$k^2 - 1$	$T_A + (1 + \lceil log_2(k-1) \rceil) T_X$

*: Type I ONB generated by irreducible all-one polynomials (AOP) exists(Reyhani-Masoleh and Hasan, 2002).

following clock cycles, xy_i is accumulated added to the contents stored in the r-bit register. The predictor for AMD codes with $t = 1$ requires nearly same overhead in area and latency as the predictor for AMD codes with $b = 1$. A detailed estimation of the overhead when using the proposed AMD codes for the protection of digit-serial multipliers in Galois fields recommended for elliptic curve cryptographic algorithms will be shown in the next section.

6.4.1 Estimation of the Hardware Overhead for the Protection of Multipliers in Galois Fields Recommended for Elliptic Curve Cryptographic Algorithms

Table 6.1 summarizes the hardware complexity for the reduced redundancy parallel (Reyhani-Masoleh and Hasan, 2002) (Row 2 and 3) and the digit-serial (Row 1) Massey-Omura multipliers in $GF(2^k)$, where r is the bit-width of the digit, C_N is the

complexity of the normal base (Reyhani-Masoleh and Hasan, 2002) and T_A, T_X are the delays due to one AND gate and one XOR gate respectively. Row 3 corresponds to the case where there is a Type I optimal normal base in $GF(2^k)$ generated by all-one polynomials (AOP). The overhead of the presented protection architecture is affected by whether Type I and Type II optimal normal base exists in $GF(2^k)$ and $GF(2^r)$. For the existence of Type I and Type II optimal normal base, please refer to (Gao, 1993).

Table 6.2 shows the overhead of the predictor and EDN of the AMD codes with $b = 1$ or $t = 1$ generated by Theorem 6.3.1 for multipliers in $GF(2^{239})$ and $GF(2^{409})$, which are among the recommended Galois fields for ecliptic curve cryptographic algorithms (SECG, 2000). There are two columns for the number of AND and XOR gates for each type of the multiplier. The left column shows the number of gates that is required to implement the original multiplier in $GF(2^k)$. This multiplier is also duplicated in the predictor. The right column shows the number of gates that is required to implement the other parts of the predictor and EDN. To estimate the area overhead, we only consider multipliers and ignore the adder, the multiplexer in $GF(2^r)$ and the r-bit register in the predictor. We consider the cases where the multiplication in $GF(2^k)$ is completed in 2, 4 or 8 clock cycles. For each case, we select r in such a way that there is an optimal normal base of Type I in $GF(2^r)$ for the purpose of minimizing the hardware complexity of multipliers in $GF(2^r)$. When computing the percentage overhead of the predictor and EDN, we assume that the area of a XOR gate is about 1.5 times of the area of a AND gate according to the data of the 45nm NANGATE library (Nangate, 2011).

The area overhead of the predictor and EDN for codes based on based on Theorem 6.3.1 is affected by the number of clock cycles required to complete the multiplication. Generally speaking, the area overhead decreases as the number of clock cycles needed

to finish one multiplication in $GF(2^k)$ increases. For the three cases shown in Table 6.2, the overall area overhead for the predictor and EDN is about $110\% \sim 160\%$.

In addition to the area overhead, the protection architectures based on the proposed AMD codes will also increase the latency the multiplier due to the longer critical path in the predictor. For example, when $k = 409$ and $r = 106$, the latency of the serial reduced-redundancy Massey-Omura multiplier is $T_A + 9T_X$. In the predictor for codes with $b = 1$ shown in Figure 6·2, the critical path contains a digit-serial multiplier in $GF(2^{409})$, a parallel multiplier in $GF(2^{106})$ and a 2-level XOR network, assuming $e_y \neq 0$. Thus the latency of the predictor is $2T_A + 19T_X$ and is twice larger than the latency of the original multiplier. To reduce the latency, optional pipeline registers can be added between the duplicated multiplier in $GF(2^k)$ and the multiplier in $GF(2^r)$ as shown by the dotted blocks in Figure 6·2. In this case the same latency as the original multiplier can be achieved for the predictor. Similar strategy can also be applied for codes with $t = 1$ (Figure 6·3).

6.5 Summary

(k, m, r) Algebraic manipulation detection (AMD) codes for the case $m = r$ and $k = br$ were introduced in (Dodis et al., 2006) and were used in (Cramer et al., 2008) for robust secret sharing scheme and for robust fuzzy extractors. In this Chapter, we presented the bounds and general constructions of AMD codes based on the generalized Reed-Muller codes. The proposed codes can provide a guaranteed level of protection against fault injection attacks even if an attacker can fully control the fault-free outputs of the device and the non-zero error patterns. The same characteristic cannot be achieved by any previously proposed protection countermeasures based on error detecting codes in the literature. As a case study, we present the protection architectures based on the AMD codes for multipliers in Galois fields recommended

for elliptic curve cryptography. The area overhead for the protection architectures is around $110\% - 160\%$. Moreover, when the predictor is pipelined, the protected multiplier has no latency penalty and can achieve the same performance as the original device.

Table 6.2: Estimation of the total area overhead of the predictor and EDN for digit-serial multipliers in $GF(2^k)$ protected by codes generated by Theorem 6.3.1

Cycles	$k = 239$[1]							$k = 409$						
	r	AND		XOR		Percentage[3]	Q_V	r	AND		XOR		Percentage[3]	Q_V
2	130[2]	62010	33800	61880	33798	154.5%	$\frac{2}{2^{130}}(\frac{4}{2^{130}})$	226[2]	> 184642	102150	> 184416	102148	155.4%	$\frac{2}{2^{226}}(\frac{4}{2^{226}})$
4	60[2]	28620	7200	28560	7198	125.2%	$\frac{2}{2^{60}}(\frac{6}{2^{60}})$	106[2]	86602	22470	86496	22468	126.0%	$\frac{2}{2^{106}}(\frac{6}{2^{106}})$
8	36[2]	17172	2592	17136	2590	115.1%	$\frac{2}{2^{36}}(\frac{10}{2^{36}})$	52[2]	42484	5406	42432	5404	112.8%	$\frac{2}{2^{52}}(\frac{10}{2^{52}})$

[1]: There exists an optimal normal base of Type II for $GF(2^r)$.

[2]: There exists an optimal normal base of Type I for $GF(2^r)$.

[3]: We assume that the area of a XOR gate is approximately 1.5 times of the area of the AND gate according to the data of the 45nm NANGATE library(Nangate, 2011).

Chapter 7

Reliable Memories Based on Nonlinear Error Correcting Codes

Linear single-error-correcting, double-error-detecting (SEC-DED) codes widely used for the design of reliable memories such as SRAMs cannot detect and can miscorrect many errors with large Hamming weights. This may be a serious disadvantage for many modern technologies when error distributions are hard to estimate and multi-bit errors are highly probable.

In the first part of the Chapter, we propose to use nonlinear SEC-DED codes to replace linear Hamming codes to improve the reliability of memories. The nonlinear SEC-DED were described in Chapter 3 and were generalized from the existing perfect nonlinear codes (Vasil'ev codes (Vasil'ev, 1962), Phelps codes (Phelps, 1983) and the codes based on one switching constructions (Etzion and Vardy, 1994)).

Multi-level cell (MLC) NAND flash memories are popular storage media because of their power efficiency and large storage density. Conventional reliable MLC NAND flash memories based on BCH codes or Reed-Solomon (RS) codes have a large number of undetectable and miscorrected errors. Moreover, standard decoders for BCH and Reed-Solomon codes cannot be easily modified to correct errors beyond their error correcting capability $t = \lfloor \frac{d-1}{2} \rfloor$, where d is the Hamming distance of the code.

In the second part of the Chapter, we propose to use nonlinear multi-error correcting codes presented in Chapter 3 as alternatives to BCH and Reed-Solomon codes

for the protection of MLC NAND flash memories. Our constructions can generate nonlinear bit-error correcting or nonbinary digit-error correcting codes with close to zero errors undetected or miscorrected for all codewords. Moreover, codes generated by the generalized Vasil'ev construction can correct some errors with multiplicities larger than t without any extra overhead in area, latency and power consumption compared to schemes where only errors with multiplicity up to t are corrected.

In both parts of the Chapter, the error correcting algorithms for the proposed nonlinear error correcting codes are shown. The reliability, area, latency and the power consumption of the encoder and the decoder of architectures based on the proposed codes are compared to those based on linear codes. The results show that using the proposed nonlinear error correcting codes for the protection of memories can largely reduce the number of errors undetected or miscorrected for all codewords at the cost of a reasonable increase in power and area ($15\% - 30\%$) compared to architectures based on linear code.

7.1 Design of Memories with Concurrent Error Detection and Correction by Nonlinear SEC-DED Codes

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 and this number is expected to reach 90% by 2011 (Moore, 2007) (Halfhill, 2005). 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 non-destructive 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 (Johnston, 2000). Similarly, decreased voltage levels for modern technologies make bit inversions more likely to occur (Eto et al., 1998).

The dangers of possible errors in memories resulting from SEUs are often mitigated with the use of linear single-error-correcting, double-error-detecting (SEC-DED) codes. 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 2^k. 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, see e.g. (Swift, 2001; Satoh et al., 2000).

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 90nm technologies – nearly 55% of the errors due to neutron radiation were multi-bit errors (Georgakos et al., 2007). Although there are mechanisms like bit interleaving (Maiz et al., 2003) 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.

Errors that are undetected (miscorrected) by some but not all of the codewords are called **conditionally detectable (miscorrected) errors**. Linear codes do not have conditionally detectable (miscorrected) errors. All errors are either 100% detected (corrected) or not detected (corrected) at all, which is bad for the detection (correction) of **repeating errors**, since an error e will always be masked (miscorrected) as long as it is masked (miscorrected) for one single message. Repeating errors can occur in many situations. In (Lisbôa et al., 2007), it was shown that transient faults lasting for more than one clock cycle are possible for new technologies. If a SEU lasts for several consecutive READ/WRITE cycles, it is possible that different messages written into the same memory cell are affected by the same error pattern. Another example of repeating errors is a hard error caused by a permanent fault in the device that is unrecoverable by re-writing. These errors may repeat themselves until the memory is replaced. For memories with repeating errors, more powerful error correcting codes are required.

In this Section we analyze the limitations of existing error correcting architectures for memories and show how nonlinear robust codes can be applied to make memories more reliable in the presence of unpredictable environments where the error distributions are unknown or not stationary. We propose several architectures based on nonlinear SEC-DED partially robust codes, i.e. extended Vasil'ev codes and extended Phelps codes, for single bit error correction in memories. These codes have fewer undetectable errors and fewer multi-bit errors which are always miscorrected while requiring a latency penalty, hardware overhead and power consumption comparable to that of the conventional linear SEC-DED codes.

7.1.1 Previous Work

Since the basic construction of SEC-DED codes was presented by Hamming in 1950 (Hamming, 1950), a number of modifications have been proposed. In (Hsiao, 1970), a class of optimal minimum odd-weight-column SEC-DED codes was constructed for better performance, cost and reliability. To further simplify the encoding and decoding complexity, the author in (Lala, 2003) proposed a coding technique requiring less ones in the parity check matrix than the code presented in (Hsiao, 1970). In (Bhattacharryya and Nandi, 1997), a hardware efficient method was proposed to construct SEC-DED-AUED systematic codes that can also detect all unidirectional errors. For protecting byte oriented memories, SEC-DED-SBD codes were proposed in (Reddy, 1978), (Chen, 1983) and (Dunning, 1985). These codes are known as single-error-correcting, double-error detecting, single-byte-error-detecting codes and are able to detect all single byte errors. SEC-DED-SBD codes that are also able to correct any odd number of erroneous bits per byte were proposed in (Penzo et al., 1995). To enhance the error correction capability of SEC-DED codes, the author in (Dutta and Touba, 2007) constructed single-error-correcting, double-error-detecting, double-adjacent-error-correcting (SEC-DED-DAEC) code by selectively avoiding certain types of linear dependencies in the parity check matrix. These codes use the same number of check bits and the similar overhead to other known SEC-DED codes and have the advantage that it can correct all adjacent double errors. In (Chen, 1996), the author constructed single-byte-error-correcting, double-byte-error-detecting codes (SBC-DBD), which can provide complete single byte error correction capabilities. In (Lala, 1978), double-error-correcting and triple-error-detecting code was proposed to correct all double bit errors. The well known Reed-Solomon code, as another example, was utilized in Hubble Space Telescope to protect 16 Mbit DRAMs manufactured by IBM (Whitaker et al., 1991).

All the codes mentioned above are linear codes. They concentrate their error detection and correction capabilities on a specific type of errors (e.g. errors with small multiplicities or belonging to the same byte). The reliability of the memory systems based on these codes can not be guaranteed when the MBU rate is high.

Some memory protection architectures based on nonlinear codes have also been proposed in the community. In (Bose, 1984), efficient single error correcting and $d(d \geq 2)$-unidirectional error detecting codes were used to protect memories. Another nonlinear error detecting code – Berger code (Berger, 1961), was used to detect unidirectional errors in flash memories. These existing protection architectures based on nonlinear codes, however, were mainly designed for unidirectional error models. In the presence of symmetric errors, the reliability of the protected memory systems can not be guaranteed.

Several constructions of nonlinear SEC-DED codes have been shown in Chapter 3. To demonstrate the advantage of using these codes to protect memories, we compare the error correction properties, the hardware overhead and the power consumption for the $(39, 32, 4)$ extended Vasil'ev code with $Q_{mc} = 0.5$ and $\omega_d = 6$ (Example 3.4.1), the $(39, 32, 4)$ extended Phelps code with $Q_{mc} = \frac{1}{16}$ and $\omega_d = 27$ (Example 3.4.3) and the linear $(39, 32, 4)$ extended Hamming code used in (Tam, 2006) to protect double data rate DIMM memory in a Virtex-II Pro device .

7.1.2 Memory Protection Architecture Based on the Extended Hamming Code

Figure 7·1 shows the general memory architecture with error correction function based on systematic error correcting codes. During a WRITE operation, the redundant bits of the code are generated by the encoder and saved in the redundant memory block. During a READ operation, the ECC 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.

For linear SEC-DED codes, the encoder performs matrix multiplication over $GF(2)$ between the k-bit data and the encoding matrix P of the selected code. The parity check matrix used to generate the $(39, 32, 4)$ extended Hamming code C in (Tam, 2006) is in standard form $H = [P|I]$, where I is the 7×7 identity matrix and

$$P = \begin{pmatrix} 01010110101010101010110101011011 \\ 10011011001100110011011001101101 \\ 11100011110000111100011110001110 \\ 00000011111111000000011111110000 \\ 00000011111111001111100000000000 \\ 11111100000000000000000000000000 \\ 11111111111111111111111111111111 \end{pmatrix}.$$

The last redundant bit of the design in (Tam, 2006) is equal to the parity of the in-

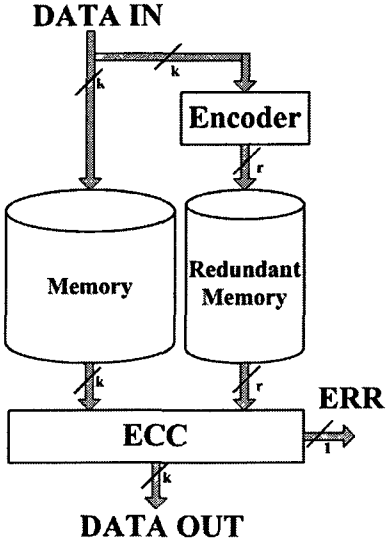


Figure 7·1: General Memory Architecture with ECC

formation bits. C is only able to detect double bit errors occurring in the information part of the code. If at least one bit of the double bit error is in the redundant portion of C, the code may miscorrect it as a single bit error. To make C a SEC-DED code, we compute the last parity bit based on all bits of the codeword.

The redundant bits are generated and written in the memory along with the associated 32-bit data. During the READ stage, the data and the redundant bits are read simultaneously. Syndromes $S = H\tilde{x}$, where $\tilde{x} \in GF(2^{39})$ is a possibly distorted output of the memory, are calculated and used to identify the error type and locate the error. A 32-bit correction mask is created to correct single bit errors occurring to the information part of the code. When a single bit error is detected, the original data is XORed with the mask and the distorted bit is reversed. When there are no errors or multi-bit errors, all the mask bits are zeros and the data go through the ECC block without any changes.

The disadvantage of memory protection architecture based on linear SEC-DED codes is the large number of undetectable and miscorrected multi-bit errors. For any linear code, $K_d = C$ and $\omega_d = k$. Thereby the number of undetectable errors for a $(39, 32, 4)$ extended Hamming code is 2^{32}. All undetectable errors correspond to distortions of more than three bits.

It is easy to prove that any (n, k, d) linear error correcting code C is able to correct not more than $2^{n-k} - 1$ errors. If N errors are corrected, $0 \leq N \leq 2^{n-k} - 1$, the number of miscorrected errors is $N(2^k - 1)$.

For example, for the approach described in (Tam, 2006), only single errors occurring to the information part of the code will be corrected. Thereby $N = 32$. The number of miscorrected multi-bit errors for this code is $32(2^{32} - 1)$.

7.1.3 Memory Protection Architecture Based on the Extended Phelps Code

Error Correction Algorithm

Let C be a $(n_1, k_1, 3)$ binary linear code and B be a $(n_2, k_2, 3)$ binary linear code. Without loss of generality, assume that $r_1 = n_1 - k_1 < r_2 = n_2 - k_2$. Denote by H_C and H_B the parity check matrix for C and B respectively. Denote by $c' =$

(x_1, x_2, x_3, x_4) a codeword of C', where $x_1 \in GF(2^{n_1}), x_2 = p(x_1) \in GF(2), x_3 \in GF(2^{n_2}), x_4 = p(x_3) \in GF(2)$, where $p(x)$ is the parity of x. Let α be a permutation of elements in $GF(2^{r_1})$ such that $\alpha(\mathbf{0}) = \mathbf{0}$. Denote by $[x_1]([x_3])$ a coset vector of the coset which $x_1(x_3)$ belongs to, $[x_1] \in GF(2^{r_1})$, $[x_3] \in GF(2^{r_2})$. The codewords of the extended Phelps code C' constructed as in Theorem 3.4.4 satisfy the condition $[x_3] = (\mathbf{0}, \alpha([x_1]))$, where $\mathbf{0} \in GF(2^{r_2-r_1})$. C' is a $(n_1 + n_2 + 2, n_1 + k_2, 4)$ SEC-DED code and is able to correct all single bit errors and simultaneously detect all double bit errors.

Denote by $e = (e_1, e_2, e_3, e_4)$ the error vector and $\tilde{c}' = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_4)$ the distorted codeword, in which $\tilde{x}_i = x_i \oplus e_i, 1 \leq i \leq 4$. The syndrome of the code that can be used to detect and locate errors are defined as $S = (S_1, S_2, S_3, S_4)$, where

$$S_1 = \tilde{x}_1, \tag{7.1}$$

$$S_2 = p(\tilde{x}_1) \oplus p(\tilde{x}_2) = p(e_1) \oplus e_2, \tag{7.2}$$

$$S_3 = \tilde{x}_3, \tag{7.3}$$

$$S_4 = p(\tilde{x}_3) \oplus p(\tilde{x}_4) = p(e_3) \oplus e_4. \tag{7.4}$$

The correction algorithm is as described below. For the purpose of comparing the error correction abilities of extended Phelps codes and linear extended Hamming codes presented before, in this algorithm only single bit errors occurring to the information part will be corrected.

1. Compute by (7.1) to (7.4) the syndrome of the code $S = (S_1, S_2, S_3, S_4)$, where $S_1 \in GF(2^{n_1}), S_3 \in GF(2^{n_2})$ and $S_2, S_4 \in GF(2)$.

2. If $S_2 = S_4 = 0$ and $[S_3] = (\mathbf{0}, \alpha([S_1]))$, no errors are detected.

3. If $S_2 = S_4 = 0$ and $[S_3] \neq (\mathbf{0}, \alpha([S_1]))$, multi-bit errors are detected and ERR will be asserted.

4. If $S_2 = 1, S_4 = 0$, multi-bit errors occur or a single bit error occurs to the first or the second part of the codeword.

   (a) If $[S_3] \geq 2^{r_1}$, multi-bit errors are detected and ERR will be asserted.

   (b) If $[S_3] < 2^{r_1}$, then

      i. If $[S_3] = (\mathbf{0}, \alpha([S_1]))$, the single bit error is in x_2.

      ii. If $[S_3] \neq (\mathbf{0}, \alpha([S_1]))$, multi-bit errors occur or a single bit error occurs to x_1. Let $\alpha^{-1} : GF(2^{r_1}) \rightarrow GF(2^{r_1})$ be the inverse function of α. Denote by $e_{[S_1]}$ and $e_{[\hat{S_1}]}$ the coset leaders of the cosets whose coset vectors are $[S_1]$ and $\alpha^{-1}([S_3]_{(r_1-1:0)})$, where $[S_3]_{(r_1-1:0)} \in GF(2^{r_1})$ is the rightmost r_1 bits of $[S_3]$. e_1 is the coset leader of the coset whose coset vector is $[e_{[S_1]} \oplus e_{[\hat{S_1}]}]$. If $||e_1|| > 1$, multi-bit errors are detected. If $||e_1|| = 1$, correct the single bit error by adding e_1 to x_1.

5. If $S_2 = 0, S_4 = 1$, multi-bit errors occur or a single bit error occurs to the third or the forth part of the codeword.

   (a) If $[S_3] = (\mathbf{0}, \alpha([S_1]))$, the single bit error is in x_4.

   (b) If $[S_3] \neq (\mathbf{0}, \alpha([S_1]))$, multi-bit errors occur or the single bit error is in x_3. Denote by $e_{[S_3]}$ and $e_{[\hat{S_3}]}$ the coset leaders of the cosets whose coset vectors are $[S_3]$ and $(\mathbf{0}, \alpha([S_1]))$. e_3 is the coset leader of the coset whose coset vector is $[e_{[S_3]} \oplus e_{[\hat{S_3}]}]$. Without loss of generality, assume the first k_2 bits of any codewords in B are the information bits. If $e_3 = e_i^*, 1 \leq i \leq k_2$, where e_i^* has one at position i and zero elsewhere, correct the single bit error by adding e_3 to x_3. If $e_3 = e_i^*, k_2 < i \leq n_2$, single bit errors occur to the redundant bits. No corrections will be attempted. If $||e_3|| > 1$, multi-bit errors are detected.

6. If $S_2 = S_4 = 1$, multi-bit errors occur. ERR will be asserted and the data will go through ECC without any correction.

Example 7.1.1 *In this example we show the error correction procedure for a* $(11, 6, 4)$ *extended Phelps code.Let* C *be a* $(4, 1, 3)$ *linear code whose parity check matrix is*

$$H_C = \begin{pmatrix} 1001 \\ 1010 \\ 0100 \end{pmatrix}.$$

Let B *be a* $(5, 2, 3)$ *linear shortened Hamming code with*

$$H_B = \begin{pmatrix} 01001 \\ 11010 \\ 10100 \end{pmatrix}.$$

Let $\{C_0 = C, C_1, \cdots, C_7\}, |C_i| = 2, 0 \leq i \leq 7$ *be a partition of* $GF(2^4)$ *into cosets of* C_0 *and* $\{B_0 = B, B_1, \cdots, B_7\}, |B_i| = 4, 0 \leq i \leq 7$ *be a partition of* $GF(2^5)$ *into cosets of* B_0. *The coset leaders and coset vectors for* C_i *and* B_i *are selected as stated in Table 7.1. It is easy to verify that* $[x \oplus y] = [x] \oplus [y]$ *is satisfied.*

Table 7.1: Selected coset leaders and coset vectors

	Coset Leader	Coset Vector
$C_0 = C$	0000	000
C_1	0001	001
C_2	0010	010
C_3	1000	011
C_4	0100	100
C_5	1110	101
C_6	0110	110
C_7	1100	111
$B_0 = B$	00000	000
B_1	00001	001
B_2	00010	010
B_3	01000	011
B_4	00100	100
B_5	11000	101
B_6	10000	110
B_7	01100	111

Let $011001 \in GF(2^6)$ *be the message that needs to be encoded.* x_1 *is the first four bits of the message:* $x_1 = 0110 \in GF(2^4)$, *which belongs to* C_6 *whose coset vector is* 110. *Let* $\alpha([x_1]) = [x_1]^3$ *where* $x_1 \in GF(2^3)$. *Select the primitive polynomial to be* $[x_1]^3 + [x_1] + 1$ *for* $GF(2^3)$. *Then* $\alpha(110) = 111$. *So* $x_3 \in B_7$. x_3 *is equal to*

the vector in B_7 *with* 01 *for the information bits, which is* 01100. $x_2 = p(x_1) = 0$, $x_4 = p(x_3) = 0$. *So the entire codeword is* $c = 01100011000 \in GF(2^{11})$. *Suppose a single bit error occurs to the* 6_{th} *bit of the codeword,* $\tilde{c} = 01100111000$. *Then* $S_1 = 0110$, $[S_1] = 110$, $S_2 = 0$, $S_3 = 11100$ *and* $S_4 = 1$. $H_B S_3 = H_B \times 00001 = 100$, *so* $S_3 = \tilde{x}_3$ *belongs to* B_1 *and* $[S_3] = 001$. *Thereby* $\alpha([S_1]) \neq [S_3]$ *and the error is in* x_3. $e_{[S_3]} \oplus e_{[\tilde{S}_3]} = 00001 \oplus 01100 = 01101$. *So the error is the coset leader of the coset that* 01101 *belongs to.* $H_B \times 01101 = H_B \times 10000$, *hence* $01101 \in B_6$ *and* $e_3 = 10000$.

Theorem 7.1.1 *Let* C *be a* $(n_1, k_1, 3)$ *binary linear code and* B *be a* $(n_2, k_2, 3)$ *binary linear code with* $r_1 = r_2$, $2^{r_1 - 1} > max\{n_1, k_2\}$, *where* $r_1 = n_1 - k_1$ *and* $r_2 = n_2 - k_2$. *Assume that* α *is an almost perfect nonlinear function with* $P_\alpha = 2^{-r_1+1}$. *The* $(n_1 + n_2 + 2, n_1 + k_2, 4)$ *extended generalized Phelps code* C' *constructed as in Theorem 3.4.4 has* $\omega_d = k_1 + k_2$ *(dimension of the detection kernel). The size of the correction kernel of the code is* $(n_1 + k_2)(2^{k_1 + k_2} - 1)$.

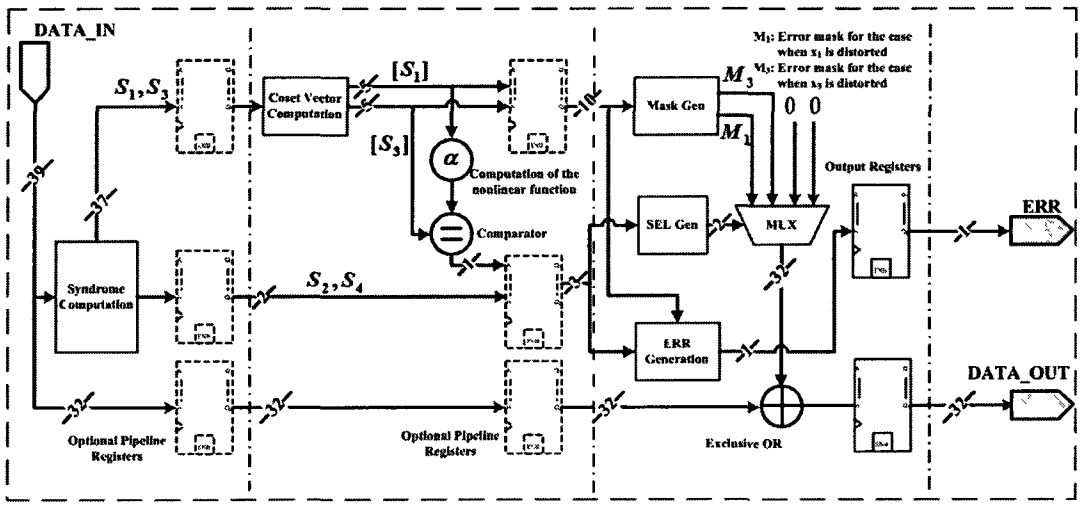


Figure 7·2: The decoder architecture for the $(39, 32, 4)$ extended Phelps code

Proof We divide the errors into four classes as stated below.

1. A nonzero error $e = (e_1, e_2, e_3, e_4)$ is masked if and only if it satisfies $S_2 = 0, S_4 = 0$ and $[S_3] = \alpha([S_1])$. $S_2 = 0$ and $S_4 = 0$ are satisfied if and only if $e_2 = p(e_1), e_4 = p(e_3)$. If $e_1 \in C$ and $e_3 \in B$, $[S_3] = \alpha([S_1])$ is always satisfied.

These errors are undetectable and form the detection kernel of the code. The number of errors in this class is $2^{k_1+k_2}$. If $e_1 \notin C$ and $e_3 \notin B$, errors will be conditionally detectable. The number of these errors is $(2^{n_1} - 2^{k_1})(2^{n_2} - 2^{k_2})$. If $e_1 \in C, e_3 \notin B$ or $e_1 \notin C, e_3 \in B$, errors will always be detected.

2. If $S_2 = 0, S_4 = 0$ and $[S_3] \neq \alpha([S_1])$, multi-bit errors are detected.

3. $S_2 = 1, S_4 = 0, [S_3] = \alpha([S_1])$. We assume that a single bit error occurs to x_2. The error will be detected but not corrected.

4. $S_2 = 1, S_4 = 0, [S_3] \neq \alpha([S_1])$. We assume that a single bit error occurs to x_1. $S_2 = 1$ and $S_4 = 0$ are satisfied if and only if $e_2 = p(e_1) \oplus 1$ and $e_4 = p(e_3)$.

   (a) If $e_1 \in C, e_3 \notin B$, $[S_3] \neq \alpha([S_1])$ is always satisfied. In this case $[x_1 \oplus e_1] = [x_1]$. A multi-bit error e is miscorrected as a single bit error $e_i^*, 1 \leq i \leq n_1$ if and only if $\alpha([x_1] \oplus [e_i^*]) = \alpha([x_1]) \oplus [e_3]$. For every e_i^*, there are at most $2^{r_1} P_\alpha$ solutions for $[x_1]$. Since $P_\alpha = 2^{-r_1+1}$ and $2^{r_1-1} > n_1$, for all n_1 possible e_i^*, the total number of solutions for $[x_1]$ satisfying $\alpha([x_1] \oplus [e_i^*]) = \alpha([x_1]) \oplus [e_3]$ is less than 2^{r_1}. Thereby errors in this class are conditionally miscorrected.

   (b) If $e_1 \notin C, e_3 \in B$, $[S_3] \neq \alpha([S_1])$ is always satisfied. In this case $[x_3 \oplus e_3] = [x_3]$. A multi-bit error e is miscorrected as a single bit error $e_i^*, 1 \leq i \leq n_1$ if and only if $\alpha([x_1] \oplus [e_1] \oplus [e_i^*]) = \alpha([x_1])$. If $[e_1] = [e_i^*]$, errors will be corrected as e_i^* for all codewords. The number of errors in this class is $n_1 2^{k_1+k_2}$. n_1 of them are successfully corrected. The other $n_1(2^{k_1+k_2} - 1)$ errors belong to K_C. If $[e_1] \neq [e_i^*]$, the error will be conditionally miscorrected.

   (c) If $e_1 \notin C$, $e_3 \notin B$. A multi-bit error e is miscorrected as a single bit error $e_i^*, 1 \leq i \leq n_1$ if and only if $\alpha([x_1] \oplus [e_1] \oplus [e_i^*]) = \alpha([x_1]) \oplus [e_3]$. If $[e_1] = [e_i^*]$,

errors will be always detected. If $[e_1] \neq [e_i^*]$, errors will be conditionally miscorrected.

5. $S_2 = 0, S_4 = 1, [S_3] = \alpha([S_1])$. We assume that a single bit error occurs to x_4. The error will be detected but not corrected.

6. $S_2 = 0, S_4 = 1, [S_3] \neq \alpha([S_1])$. We assume that a single bit error occurs to x_3.

   (a) If $e_1 \in C, e_3 \notin B$, $[S_3] \neq \alpha([S_1])$ is always satisfied. In this case a multi-bit error will be miscorrected as $e_i^*, 1 \leq i \leq k_2$ occurring to x_3 if and only if $\alpha([x_1]) = \alpha([x_1]) \oplus [e_3] \oplus [e_i^*]$. Errors will be corrected as $[e_i^*]$ by all codewords if $[e_3] = [e_i^*]$. The number of these errors is $k_2 2^{k_1+k_2}$. k_2 of them are successfully corrected. The other $k_2(2^{k_1+k_2} - 1)$ belong to K_C. If $[e_3] \neq [e_i^*]$, errors are always detected.

   (b) If $e_1 \notin C, e_3 \in B$, $[S_3] \neq \alpha([S_1])$ is always satisfied. In this case a multi-bit error will be miscorrected as $e_i^*, 1 \leq i \leq k_2$ occurring to x_3 if and only if $\alpha([x_1] \oplus [e_1]) = \alpha([x_1]) \oplus [e_i^*]$. Following the same analysis as for $4.(a)$, errors in this class will be conditionally miscorrected.

   (c) If $e_1 \notin C$, $e_3 \notin B$, a multi-bit error will be miscorrected as $e_i^*, 1 \leq i \leq k_2$ occurring to x_3 if and only if $\alpha([x_1] \oplus [e_1]) = \alpha([x_1]) \oplus [e_3] \oplus [e_i^*]$. These errors will be conditionally miscorrected.

7. $S_2 = 1, S_4 = 1$. In this case multi-bit errors occur and no error correction will be attempted. The number of errors in this class is $2^{n_1+n_2}$. ■

From Theorem 7.1.1, it is easy to show that the $(39, 32, 4)$ extended Phelps code constructed in Example 3.4.3 has $\omega_d = k_1 + k_2 = 5 + 22 = 27$. The size of the correction kernel is $(n_1 + k_2)(2^{k_1+k_2} - 1) = 32(2^{27} - 1)$.

Hardware Implementation of the Encoder and the Decoder for the Extended Phelps Code

The encoder for the extended Phelps code is mainly composed of the following parts:

1. Syndrome computation unit for C;
2. Circuits to realize the nonlinear permutation α. In our case α is the cube operation in $GF(2^{r_1})$.
3. Encoder for the linear Hamming code B;
4. Exclusive OR network to convert codewords of B to vectors in other cosets;
5. Parity check generation unit.

The input to the encoder can be any (n_1+k_2)-bit binary vector. The first n_1-bit is x_1 and the left k_2-bit is the information part of x_3. The syndrome computation unit computes $H_C x_1$, which is used to determine the coset that x_1 belongs to. $[x_3] = [x_1]^3$. x_3 is computed by first derive the codeword in B and then mask it with the coset leader of the coset which x_3 belongs to. The parity check generation unit computes the parity bits x_2 and x_4.

The architecture of the decoder for the extended Phelps code is shown in Figure 7·2. After receiving a possibly distorted codeword $\tilde{c}' = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_4)$, the syndrome $S = (S_1, S_2, S_3, S_4)$ will be computed. S_1 and S_3 are used to determine the coset vectors of the coset which $\tilde{x}_1$ and $\tilde{x}_3$ belong to. Then whether $[S_3] = [S_1]^3$ will be tested. To speed up the design, the possible error vector e_1 and e_3 will be precomputed and sent to a MUX before the type of the error is known. If a single bit error in x_1 or the first k_2 bits of x_3 is detected, e_1 or e_3 will be XORed with the corresponding part of the received data to recover the original message. If multi-

bit errors are detected, the ERR generation unit will pull up the ERR signal. The received data will go through the ECC module without any error correction.

The latency penalty, the hardware overhead and the power consumption of the encoder and the non-pipelined decoder for the extended Phelps code will be shown in section 7.1.5. We also note that the design of the decoder for the extended Phelps code can be pipelined to increase the throughput of the system. The possible locations of the pipeline registers are shown in Figure 7·2.

7.1.4 Memory Protection Architecture Based on the Extended Vasil'ev Code

Error Correction Algorithm

The codewords of a $(a+m+2, a+k_V, 4)$ extended Vasil'ev code constructed as in Theorem 3.4.2 are in the format of

$$(u, (u, \mathbf{0}) \oplus v, p(u) \oplus f(y), p(u) \oplus p(v) \oplus f(y)),$$

where $u \in GF(2^a)$, $\mathbf{0} \in GF(2^{m-a}), 0 < a \leq m$, $v \in V$ is the codeword of a $(m, k_V, 3)$ Hamming code V, $y \in GF(2^{k_V})$ are the information bits of v, $f : GF(2^{k_V}) \rightarrow \{0,1\}$ is a nonlinear mapping satisfying $f(\mathbf{0}) = 0$ and p is the linear parity function. In order to simplify the encoding and decoding complexities, we select V to be a linear Hamming code.

The redundant portion of the extended Vasil'ev code contains three parts. The first part is the redundant bits of V which can be generated by a linear XOR network performing matrix multiplication over $GF(2)$. The second and the third part are non-linear. The encoder for these two parts needs to perform the linear parity predictions $p(u), p(v)$ as well as the nonlinear mapping $f : GF(2^{k_V}) \rightarrow \{0,1\}$. When k_V is even, we can select f to be the non-repetitive quadratic function (Example 3.4.5) for the

purpose of minimizing Q_{mc}.

$$f(v) = v_1 \cdot v_2 \oplus v_3 \cdot v_4 \oplus v_5 \cdot v_6 \oplus \cdots \oplus v_{k_V-3} \cdot v_{k_V-2} \oplus v_{k_V-1} \cdot v_{k_V}. \quad (7.5)$$

Before we describe the error correction algorithm for the extended Vasil'ev code, the syndrome S for locating and correcting errors need to be defined. Denote by $c = (x_1, x_2, x_3, x_4)$ a codeword of the extended Vasil'ev code, $e = (e_1, e_2, e_3, e_4)$ an error vector and $\tilde{c} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_4)$ the distorted codeword. Then

$$\begin{aligned} x_1 &= u, \\ x_2 &= (u, \mathbf{0}) \oplus v, \\ x_3 &= p(u) \oplus f(y), \\ x_4 &= p(u) \oplus p(v) \oplus f(y). \end{aligned}$$

Let H be the parity check matrix of the linear code V and $\tilde{y}$ be the distorted information bits of V. The syndrome can be defined as $S = (S_1, S_2, S_3)$, where

$$S_1 = H((\tilde{x}_1, \mathbf{0}) \oplus \tilde{x}_2), \quad (7.6)$$

$$S_2 = p(\tilde{x}_1) \oplus f(\tilde{y}) \oplus \tilde{x}_3, \quad (7.7)$$

$$S_3 = p(\tilde{x}_1) \oplus p(\tilde{x}_2) \oplus p(\tilde{x}_3) \oplus p(\tilde{x}_4). \quad (7.8)$$

The error correction algorithm is as stated below. Similar to the design described in (Tam, 2006), only single errors in the information part of the code will be corrected. If single errors are in the redundant portion or multi-bit errors are detected, ERR will be asserted but no correction will be attempted.

1. Compute by (7.6),(7.7),(7.8) the syndrome $S = (S_1, S_2, S_3)$ of the code, where $S_1 \in GF(2^{\lceil log_2(m+1) \rceil})$ and $S_2, S_3 \in GF(2)$.
2. If S is the all zero vector, then no error is detected. Otherwise one or more

errors are detected.

3. If $S_3 = 0$ and at least one of S_1, S_2 is nonzero, errors with even multiplicities are detected and ERR will be raised. Errors in this class are uncorrectable because all of them are multi-bit errors.
4. If $S_3 = 1$ and $S_1 = \mathbf{0}$, a single bit error occurs to one of the last two redundant bits of the code. ERR will be asserted and the data will go through ECC without any correction because only single bit errors in the information part need to be corrected.
5. If $S_3 = 1, S_1 \neq \mathbf{0}$ and S_1 does not match any columns of the parity check matrix H, an uncorrectable multi-bit error of an odd multiplicity is detected and ERR will be raised.
6. If $S_3 = 1$ and $S_1 = h_i$, where h_i is the i_{th} column of H, a single bit error in the first two parts of the code or multi-bit errors are detected. Without loss of generality, we assume that the first $k_V = m - \lceil log_2(m+1) \rceil$ bits of V are information bits.
   (a) If $a \leq k_V$ and $1 \leq i \leq a$, flip the i_{th} bit of x_1, recalculate S_2. If $S_2 = 0$, the single error is at the i_{th} bit of x_1 and is successfully corrected. Otherwise the single error is at the i_{th} bit of x_2.
   (b) If $a \leq k_V$ and $a < i \leq k_V$ flip the i_{th} bit of x_2, recalculate S_2. If $S_2 = 0$, the single error is in the i_{th} bit of x_2 and is successfully corrected. Otherwise multi-bit errors with odd multiplicities are detected.
   (c) If $a \leq k_V$ and $i > k_V$, the error occurs to the redundant bits of V and does not need to be corrected. ERR will be asserted and no correction will be attempted.
   (d) Similar procedures can be applied to the case when $a > k_V$.

Example 7.1.2 *In this example we show the encoding and decoding procedure for a*

$(39, 32, 4)$ extended Vasil'ev code C with $a = 6$. When $a = 6$, $u \in GF(2^6), v = (y, z) \in GF(2^{31}), k_V = 26$, where $y \in GF(2^{26})$ are the information bits and $z \in GF(2^5)$ are the redundant bits of the $(31, 26, 3)$ binary Hamming code. Let

$$11111001011011000110010111001111 \in GF(2^{32})$$

be the message that needs to be encoded. Then $u = 111110$ and $p(u) = 1$. y can be computed by XOR $(u, \mathbf{0}), \mathbf{0} \in GF(2^{|a-k_V|})$ with the other $k_V = 26$ bits of the message. Thus

$$y = 10100011000110010111001111 \in GF(2^{26}).$$

Let

$$H = \begin{pmatrix} 11111011101101001111000000010000 \\ 11110111011010101000111000001000 \\ 11101110110110010100100110000100 \\ 11011101110001110010010100100010 \\ 10111100001111110001001011000001 \end{pmatrix}$$

be the parity check matrix of V. Then the redundant bits z of $v \in V$ are 00101 and $p(v) = 0$. Let f be the non-repetitive quadratic function as described in Example 3.4.5, then $f(y) = 0$. The last two nonlinear redundant bits are 11. The entire codeword is

$$111110010110110001100101110011110010111 \in GF(2^{39}).$$

Suppose a single bit error occurs to the 9_{th} bit of the codeword. After receiving the distorted codeword, S_1, S_2 and S_3 can be computed according to (7.6),(7.7),(7.8). We have $S_1 = h_3 = 11101, S_2 = 0, S_3 = 1$. The 3_{rd} bit of x_1 is flipped and S_2 is recomputed. The new value of S_2 is one. So the error is at the 3_{rd} bit of the x_2, i.e. 9_{th} bit of the entire codeword.

The sizes of K_d and K_c for the extended Vasil'ev code can be computed according to the next theorem.

Theorem 7.1.2 *For $(a + m + 2, a + k_V, 4)$ extended Vasil'ev codes, where $k_V = m - \lceil log_2(m + 1) \rceil$, let $t = min\{a, k_V\}$, there are 2^a undetectable errors and $2^{a+1}(2^{k_V} - 1)$ conditionally detectable errors. If only errors occurring to the information part of the code are corrected, the number of miscorrected errors is $2t(2^{a+k_V} - 1) + (2^t - 1)|a - k_V|$. The number of conditionally miscorrected errors is $2|a - k_V|(2^{a+k_V} - 2^t)$. The probability of error masking for conditionally detectable errors and the probability of*

miscorrection for conditionally miscorrected errors are bounded by P_f, which is the nonlinearity of f defined by (4).

Proof The syndrome of the code can be re-written as follows.

$$\begin{aligned} S_1 &= H((\tilde{x}_1, \mathbf{0}) \oplus \tilde{x}_2) = H((e_1, \mathbf{0}) \oplus e_2), \\ S_2 &= p(\tilde{x}_1) \oplus f(\tilde{y}) \oplus \tilde{x}_3 \\ &= f(\tilde{y}) \oplus f(y) \oplus p(e_1) \oplus e_3, \\ S_3 &= p(\tilde{x}_1) \oplus p(\tilde{x}_2) \oplus p(\tilde{x}_3) \oplus p(\tilde{x}_4) \\ &= p(e_1) \oplus p(e_2) \oplus p(e_3) \oplus p(e_4). \end{aligned}$$

1. $K_d = \{e | S_1 = \mathbf{0} \in GF(2^{\lceil log_2(m+1) \rceil}), S_2 = S_3 = 0 \in GF(2), \forall c \in C\}$. Since $S_1 = H((e_1, \mathbf{0}) \oplus e_2) = \mathbf{0}$, $(e_1, \mathbf{0}) \oplus e_2$ is a codeword of the linear code V. Because f is a nonlinear function, the only possibility to guarantee $S_2 = 0, \forall c \in C$ is that $(e_1, \mathbf{0}) = e_2, p(e_1) = e_3$. $S_3 = 0$, thus $e_4 = e_3 = p(e_1)$. So the detection kernel of the code contains all error vectors $e = (e_1, e_2, e_3, e_4)$ such that $(e_1, \mathbf{0}) = e_2, e_3 = e_4 = p(e_1)$. The number of errors in this class is 2^a;
2. If $(e_1, \mathbf{0}) \oplus e_2$ is a nonzero codeword of V and $e_4 = p(e_1) \oplus p(e_2) \oplus p(e_3)$, then $S_1 = \mathbf{0}, S_3 = 0, \forall c \in C$. S_2 can be either one or zero depending on the information part of the code. These errors will be conditionally detected. The error masking probability is bounded by P_f. If f is a perfect nonlinear Boolean function, these errors will be detected with probability 0.5. The number of errors in this class is $2^{a+1}(2^{k_V} - 1)$.
3. Multi-bit error e will be miscorrected as a single bit error occurring to the information part of the code if and only if $S_3 = 1, S_1 = h_i, 1 \leq i \leq max\{a, k_V\}$. Let $t = min\{a, k_V\}$.
   (a) If $1 \leq i \leq t$, e will always be miscorrected as a single error in the i_{th} bit of

either x_1 or x_2. The number of pairs of e_1, e_2 satisfying $S_1 = H((e_1, \mathbf{0}) \oplus e_2) = h_i, 1 \leq i \leq t$ is $t \times 2^{a+k_V}$. e_3 can be either one or zero. $e_4 = p(e_1) \oplus p(e_2) \oplus p(e_3) \oplus 1$. So there are $2t \times 2^{a+k_V}$ errors that satisfy $S_3 = 1, S_1 = h_i, 1 \leq i \leq t$. $2t$ of them are correctly corrected. The number of miscorrected errors in this class is $2t(2^{a+k_V} - 1)$.

(b) If $t < i \leq max\{a, k_V\}$, the number of errors satisfying $S_3 = 1, S_1 = h_i$ is $2|a - k_V| \times 2^{a+k_V}$. After flipping the i_{th} bit of either $\tilde{x}_1$ or $\tilde{x}_2$, S_1 and S_3 become zero. Denote by $\hat{e}_1, \hat{e}_2$ the new error vectors after flipping the bit for the first two parts of the codewords.

i. If $(\hat{e}_1, \mathbf{0}) = \hat{e}_2$ and $e_3 = p(\hat{e}_1)$, S_2 is always zero. The number of errors in this class is $2^t \times |a - k_V|$ and $|a - k_V|$ of them are correctly corrected. The number of miscorrected errors is $(2^t - 1)|a - k_V|$.

ii. If $(\hat{e}_1, \mathbf{0}) = \hat{e}_2$ and $e_3 \neq p(\hat{e}_1)$, S_2 is always one. Errors in this class are always detectable. The number of them is $2^t|a - k_V|$.

iii. If $(\hat{e}_1, \mathbf{0}) \neq \hat{e}_2$, then S_2 can be either one or zero depending on the information bits of the code. Errors in this class will be conditionally miscorrected. The probability of miscorrection is bounded by P_f. If f is a perfect nonlinear function, the probability of miscorrection is 0.5. The number of errors in this class is $2|a - k_V|(2^{a+k_V} - 2^t)$. ■

The sizes of the detection and the correction kernel are functions of a and m. For any extended Vasil'ev codes with length n and number of information bits k, we have

$$\begin{aligned} k &= a + k_V = a + m - \lceil log_2(m+1) \rceil, \\ n &= a + m + 2, \\ a &\leq m. \end{aligned}$$

Hence $n - 2^{n-k-2} - 1 \leq a \leq \lfloor \frac{n-2}{2} \rfloor$. When $n = 39, k = 32$, $6 \leq a \leq 18$. Figure 7·3 shows the sizes of the detection and the correction kernel for $(39, 32, 4)$ extended Vasil'ev codes for different a. The minimum sizes of the detection and the correction kernel are 2^6 and $12(2^{32} - 1) + 20(2^6 - 1)$ respectively, both of which are achieved when $a = 6$. Different from traditional linear error detecting codes, extended Vasil'ev

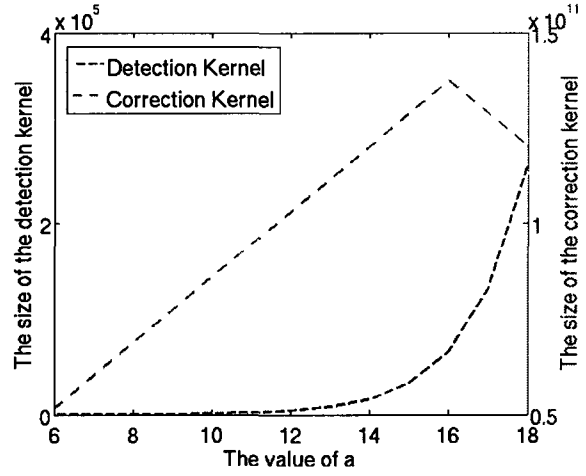


Figure 7·3: Kernels of $(39, 32, 4)$ extended Vasil'ev codes as a function of "a"

codes have conditionally undetectable or miscorrected errors. For $(39, 32, 4)$ extended Vasil'ev code with $a = 6$, the numbers of errors which are masked or miscorrected with probability 0.5 are $2(2^{32} - 2^6)$ and $40(2^{32} - 2^6)$ respectively.

Hardware Implementation of the Encoder and the Decoder for the Extended Vasil'ev Codes

The encoder for the $(39, 32, 4)$ extended Vasil'ev code is similar to the encoder for the $(39, 32, 4)$ extended Hamming code. The main difference between them is that the encoder for the extended Vasil'ev code needs to realize one nonlinear function f (see (7.5)), which requires thirteen 2-input AND gates and twelve 2-input XOR gates given the fact that $k_V = 26$. The latency penalty, the hardware overhead and the

power consumption of the encoder for the extended Vasil'ev code is comparable to that of the linear extended Hamming code (Table 7.4).

The architecture of the decoder for the extended Vasil'ev code is shown in Figure 7·4. After new data arrives, the syndrome $S = (S_1, S_2, S_3)$ is computed. If a single error in the information part is detected, we first assume that the error is in x_2 and attempt to correct the error by flipping the erroneous bit in x_2. Then S_2 will be re-computed. If S_2 becomes zero, then the error is successfully corrected. Otherwise the error is in x_1. The bit in x_2 will be flipped back and the distorted bit in x_1 will be corrected, which is done by masking the output of the first XOR network with a mask pre-computed according to the value of the syndrome.

Pipeline registers can be added to increase the throughput of the system. Possible locations for the pipeline registers are shown in Figure 7·4. In section 7.1.5, the latency penalty, the hardware overhead and the power consumption of the non-pipelined decoder for the extended Vasil'ev code will be shown and compared to that of the other two alternatives.

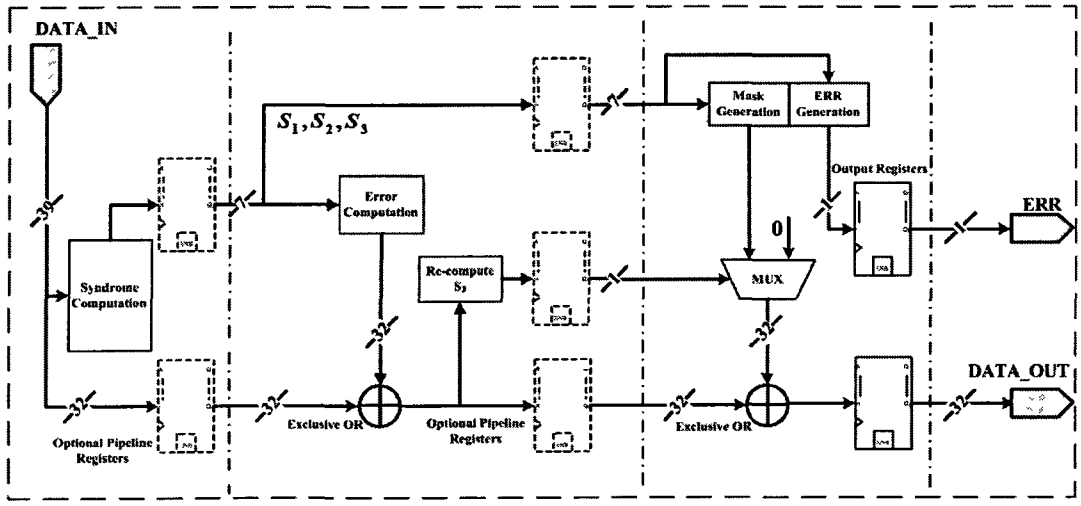


Figure 7·4: The decoder architecture for the $(39, 32, 4)$ extended Vasil'ev code

7.1.5 Comparison of Memory Architectures Based on Extended Hamming Codes, Extended Vasil'ev Codes and Extended Phelps Codes

Error Detection and Correction Capabilities

In Table 7.2 we summarize the size of the detection kernel K_d, the size of the correction kernel K_C and the worst case error masking probability Q_{mc} for errors outside the detection kernel for the $(39, 32, 4)$ extended Hamming code, the $(39, 32, 4)$ extended Vasil'ev code with $\omega = 6, Q_{mc} = 0.5$ (Example 3.4.1) and the $(39, 32, 4)$ extended Phelps code with $\omega_d = 27, Q_{mc} = \frac{1}{16}$ (Example 3.4.3). The extended Hamming code has the largest size of K_d and the largest size of K_C among the three alternatives. The extended Vasil'ev code has only 2^6 undetectable errors. The size of K_C for the extended Vasil'ev code is nearly one third of that for the extended Hamming code. The extended Phelps code has the smallest Q_{mc} and the smallest size of K_C. Its K_d is larger than that of the extended Vasil'ev code but is only $\frac{1}{32}$ of the size of K_d for the extended Hamming code.

Table 7.2: Detection and correction kernels for the $(39, 32, 4)$ extended Hamming code, the $(39, 32, 4)$ extended Vasil'ev code with $\omega_d = 6, Q_{mc} = 0.5$ and the $(39, 32, 4)$ extended Phelps code with $\omega_d = 27, Q_{mc} = \frac{1}{16}$

	Q_{mc}	size of K_d	size of K_C
Hamming	1	2^{32}	$32 \cdot 2^{32} - 1$
Vasil'ev	0.5	2^6	$\approx 12 \cdot (2^{32} - 1)$
Phelps	$\frac{1}{16}$	2^{27}	$32 \cdot (2^{27} - 1)$

Table 7.3 shows the number of undetectable and miscorrected errors with multiplicities three to six for the three alternatives. All the three codes have no undetectable single, double and triple errors. Only errors with odd multiplicities are miscorrected. The total number of miscorrected errors with multiplicities less or equal to six for the extended Vasil'ev code is less than one half of the corresponding

number for the extended Hamming code. The number of miscorrected errors with multiplicity three to six for the extended Phelps code is the smallest of the three, which makes it a good choice for error correction purpose.

We note that errors $e = (e_1, e_2, e_3, e_4)$ in the detection kernel of the extended Phelps codes satisfy

$$e_1 \in C, e_2 = p(e_1), e_3 \in B, e_4 = p(e_3).$$

Errors in the correction kernel of the extended Phelps codes satisfy either

$$[e_1] = [e_i^*], e_2 = p(e_1) \oplus 1, e_3 \in B, e_4 = p(e_3),$$

or

$$e_1 \in C, e_2 = p(e_1), [e_3] = [e_i^*], e_4 = p(e_3) \oplus 1.$$

Obviously, the number of undetectable triple errors and miscorrected quadruple errors of the extended Phelps codes can be further reduced by minimizing the number of codewords of Hamming weight three and four in codes C and B.

Table 7.3: Number of undetectable and miscorrected errors with multiplicities less or equal to six for the $(39, 32, 4)$ extended Hamming code, the $(39, 32, 4)$ extended Vasil'ev code with $\omega_d = 6, Q_{mc} = 0.5$ and the $(39, 32, 4)$ extended Phelps code with $\omega_d = 27, Q_{mc} = \frac{1}{16}$

	Code	$\|\|e\|\| = 3$	$\|\|e\|\| = 4$	$\|\|e\|\| = 5$	$\|\|e\|\| = 6$
$\|K_d\|$	Ext. Hamming	0	1583	0	51744
	Ext. Vasil'ev	0	21	0	0
	Ext. Phelps	0	364	0	3362
$\|K_c\|$	Ext. Hamming	5176	0	254432	0
	Ext. Vasil'ev	1635	0	108993	0
	Ext. Phelps	2263	0	42692	0

Area Overhead, Power Consumption and Latency

The encoder and the non-pipelined decoder for all the three codes have been modeled in Verilog and synthesized in Cadence Encounter RTL Compiler using the Nangate 45nm Opencell library (Nangate, 2011). The designs were placed and routed using Cadence Encounter.

The latency, area overhead and the power consumption of the encoders and decoders for the memory protection architectures based on the three alternatives are shown in Table 7.4 and 7.5. The data is for the typical operation condition assuming a supply voltage of $1.1V$ and a temperature of $25C$.

The encoder for the extended Vasil'ev code has almost the same area overhead and power consumption as that of the linear extended Hamming code. The decoder for the extended Vasil'ev code requires 23% more area overhead and consumes 17.15% more power than that of the extended Hamming code. In terms of the latency, the architectures based on the extended Vasil'ev code results in 21.2% more latency for the decoder and 26.5% more latency for the encoder compared to the extended Hamming code. We note that the output of the memory can be directly forwarded to the processors before it goes through the decoder. When the decoder detects or corrects errors, the processor can be stalled and the data can be re-fetched from either the output of the decoder (error correction) or the memory (error detection). In this case the latency penalty for the decoder, which is the main factor that affects the performance of the memory system, is only incurred when there are errors.

For both the encoder and the decoder, the Phelps code has the largest penalty in all the three aspects. Compared to the extended Hamming code, the latency penalty of the extended Phelps code is increased by 47.9% for the encoder and increased by 24.5% for the decoder. The area overhead and the power consumption of the decoder for the extended Phelps code is almost three times of that for the extended Hamming

Table 7.4: Latency, area overhead and power consumption for the encoders for $(39, 32, 4)$ SEC-DED codes (Voltage = 1.1 V, Temperature = 25 C)

	Latency (ns)	Area (μm^2)	Power (mW)
extended Hamming	0.290	282.2	0.2898
extended Vasil'ev	0.367	296.1	0.2916
extended Phelps	0.429	383.0	0.4728

code. However, for reliable memory systems the area and power consumption overhead is mainly contributed by the redundant memory cells, which is determined by the number of redundant bits of the error correcting code. Since all the three alternatives have the same number of redundant bits, they have similar area and power consumption overhead.

Thereby, compared to the extended Hamming code, the extended Vasil'ev code and the extended Phelps code can achieve better error detection and correction capabilities at the cost of only a small penalty in latency, area overhead and the power consumption.

Table 7.5: Latency, area overhead and power consumption for the decoders for $(39, 32, 4)$ SEC-DED codes (Voltage = 1.1 V, Temperature = 25 C)

	Latency (ns)	Area (μm^2)	Power (mW)
extended Hamming	0.538	620.3	0.7119
extended Vasil'ev	0.652	763.2	0.8340
extended Phelps	0.670	1799.8	1.774

7.1.6 Further Discussions

We note that errors in the decoders for the proposed nonlinear SEC-DED codes may also compromise the reliability of the memories. To protect the decoder against errors caused by temporary faults like SEU and MBU, a *fault secure* design is required. A circuit is fault secure for a fault set ϵ if every fault in ϵ can be detected as long as

it manifests at the output of the circuit. The design of fault secure circuits is well studied in the community (see e.g. (Rao and Fujiwara, 1989)). Most of the existing technologies of designing fault secure circuits (e.g. methodologies based on duplication and two-rail code checker (Rao and Fujiwara, 1989)) can be directly applied to build fault secure decoders for the nonlinear SEC-DED codes.

7.2 Reliable MLC NAND Flash Memories Based on Nonlinear Multi-Error Correction Codes

The semiconductor industry has witnessed an explosive growth of the NAND flash memory market in the past several decades. Due to its high data transfer rate, low power consumption, large storage density and long mechanical durability, the NAND flash memories are widely used as storage media for devices such as portable media players, digital cameras, cell phones and low-end netbooks.

The increase of the storage density and the reduction of the cost per bit of flash memories were traditionally achieved by the aggressive scaling of the memory cell transistor until the multi-level cell (MLC) technology was developed and implemented in 1997 (Atwood et al., 1997). MLC technology is based on the ability to precisely control the amount of charge stored into the floating gate of the memory cell for the purpose of setting the threshold voltage to a number of different levels corresponding to different logic values, which enables the storage of multiple bits per cell.

However, the increased number of programming threshold voltage levels has a negative impact on the reliability of the device due to the reduced operational margin. The raw bit error rate of the MLC NAND flash memory is around 10^{-6} (Cooke, 2007) and is at least two orders of magnitude worse than that of the single-level cell (SLC) NAND flash memory (Dan and Singer, 2003). Moreover, the same reliability concerns as for SLC NAND flash memories, e.g. program/read disturb, data retention,

programming/erasing endurance (Bez et al., 2003) and soft errors (Cellere et al., 2009)(Bagatin et al., 2007)(Irom and Nguyen, 2007), may become more significant for MLC NAND flash memories. Hence a powerful error correcting code (ECC) that is able to correct at least 4-bit errors is required for the MLC NAND flash memories to achieve an acceptable application bit error rate, which is no larger than 10^{-11} (Cooke, 2007).

Several works have investigated the use of linear block codes to improve the reliability of MLC NAND flash memories. In (Liu et al., 2006), the authors presented a high-throughput and low-power ECC architecture based on $(n = 4148, k = 4096, d = 9)$ BCH codes correcting quadruple errors ($t = 4$). In (Micheloni et al., 2006), a 4Gb 2b/cell NAND flash memory chip incorporating a 250MHz BCH error correcting architecture was shown. The author of (Sun et al., 2007) demonstrated that the use of strong BCH codes (e.g. $t = 12, 15, 67, 102$) can effectively increase the number of bits/cell thus further increasing the storage capacity of MLC NAND flash memories. In (Chen et al., 2009), an adaptive-rate ECC architecture based on BCH codes was proposed. The design had four operation modes with different error correcting capabilities. An ECC architecture based on Reed-Solomon codes of length 828 and 820 information digits constructed over $GF(2^{10})$ was proposed in (Chen et al., 2008), which can correct all bit errors of multiplicity less than or equal to four. The architecture achieves higher throughput, requires less area overhead for the encoder and the decoder but needs 32 more redundant bits than architectures based on BCH codes with the same error correcting capability. In (Cassuto et al., 2007), an architecture based on asymmetric limited-magnitude error correcting code was proposed, which can correct all asymmetric errors of multiplicities up to t.

The above architectures are based on linear block codes and have a large number of undetectable errors. For any linear code with k information bits, the number of

undetectable errors is 2^k, which is a potential threat to the reliability of the memory systems. The situation becomes even worse due to the possible miscorrection of errors. Let us denote a binary error vector by e and the multiplicity of the error by $||e||$. A multi-bit error $e, ||e|| > t$ is miscorrected by a linear t-error-correcting code if and only if it has the same syndrome as some e', where $||e'|| \leq t$. It is easy to show that the number of errors miscorrected for all codewords of a (n, k, d) linear t-error-correcting code is $\sum_{i=1}^{t} \binom{n}{i} \times (2^k - 1)$.

Under the assumption that errors are independent whose distribution satisfies $P(e) = \theta^{||e||}(1-\theta)^{n-||e||}$, where θ is the raw bit distortion rate and $P(e)$ is the probability of the occurrence of event e, the most harmful miscorrected errors are errors of multiplicity $t+1$. Let us denote the number of codewords of Hamming weight $2t+1$ for a linear error correcting code by A_{2t+1}. The number of errors of multiplicity $t+1$ that are miscorrected for all codewords of a linear bit-error correcting code is $\binom{2t+1}{t} \times A_{2t+1}$. For the commonly used $(n = 8262, k = 8192, d = 11)$ linear BCH codes with $t = 5$, the number of errors of multiplicity six miscorrected for all codewords is as large as $462 \times A_{11} \approx 10^{17}$. This large number of miscorrected errors of multiplicity six cannot be neglected and should be taken into account when designing reliable MLC NAND flash memories.

To reduce the number of undetectable and miscorrected errors, nonlinear *minimum distance robust* and *minimum distance partially robust* codes have been proposed in (Karpovsky and Taubin, 2004; Wang et al., 2009c; Kulikowski et al., 2008b). An ECC architecture based on nonlinear single-error-correcting, double-error-detecting (SEC-DED) codes for the protection of memories against soft errors was shown in (Wang et al., 2009c).

In this Section we propose to use the two nonlinear multi-error correcting codes presented in Chapter 3 to improve the reliability of MLC NAND flash memories. We

present the error correcting algorithms for both codes. The error correcting algorithm for the generalized Vasil'ev codes can also correct some errors beyond the error correcting capability t without any modifications and extra requirements. When $p = 2^m$, the presented constructions can also generate nonlinear error correcting codes with the same digit-error and burst-error correcting capabilities as linear Reed-Solomon codes in $GF(2^m)$. In addition to errors that are undetectable or miscorrected for all codewords, there are also some errors which are masked or miscorrected by a fraction of codewords of the presented nonlinear multi-error correcting codes. These errors are called **conditionally detectable** and **conditionally miscorrected** errors. We note that the data-dependent error detecting and correcting properties of the presented codes are useful for detecting and locating repeating errors, e.g. errors introduced by hardware malfunctions such as data retention and programming/erasing failure.

We propose ECC architectures for MLC NAND flash memories based on the presented nonlinear multi-error correcting codes. The proposed architectures have nearly no undetectable errors and errors miscorrected for all codewords at the cost of less than 20% increase in area and power consumption compared to architectures based on the widely used BCH and Reed-Solomon codes with the same bit error correcting capability t. Moreover, the reliability of the memories protected by the generalized Vasil'ev codes can be further improved given the fact that the proposed architecture is able to correct some errors of multiplicity larger than t.

7.2.1 MLC NAND Flash Memories

Multi-level cell is able to store multiple bits by precisely controlling the threshold voltage level of the cell. In practice, the threshold voltage of the whole memory array satisfies a Gaussian distribution due to random manufacturing variations (Chen et al., 2008). Figure 7·5 illustrates the threshold voltage distribution of a multi-level cell

which can store 2 bits. Let us denote the standard deviation of the middle two Gaussian distributions in Figure 7·5 by σ. The standard deviations of the outer two distributions are approximately 4σ and 2σ (Chen et al., 2008). Each voltage range corresponds to a specific logic value represented as a 2-bit binary vector. Different schemes can be used for mapping the logic values to binary vectors. A direct mapping was used in (Atwood et al., 1997). The authors in (Chen et al., 2008) proposed to use a Gray mapping to improve the reliability of the memory. If an error occurs during a READ operation, it is more likely that the original 2-bit binary vector is distorted into another 2-bit vector corresponding to the adjacent voltage level (Figure 7·5). Hence the Gray mapping can efficiently reduce the average error multiplicity thus increasing the error detecting probability compared to the direct mapping scheme.

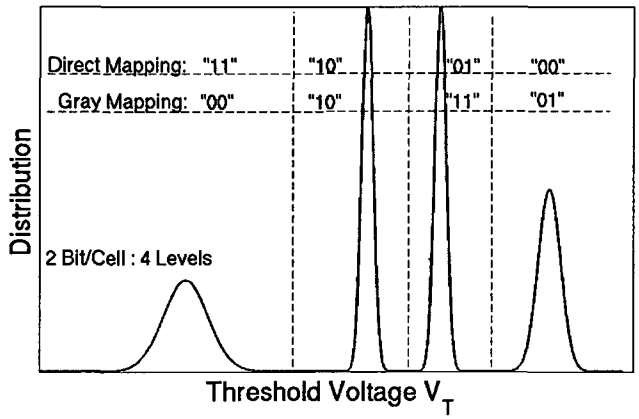


Figure 7·5: Threshold voltage distribution for a MLC storing 2 bits (Chen et al., 2008)

The data of the NAND flash memory is organized in *blocks*. Each block consists of a number of *pages*. Each page stores K *data bytes* and R *spare bytes*. Cells in the spare area are physically the same as cells in the rest of the page and are typically used for overhead functions such as ECC and wear-leveling (Micron, 2008). The proportion of the spare bytes in the total number of bytes per page is usually 3% , e.g. 64 spare bytes for 2048 data bytes. More spare bytes may be required as the page

size increases, e.g. 218 spare bytes for 4096 data bytes (Cooke, 2007). Due to the existence of spare bytes, the number of redundant bits of the error correcting codes used for NAND flash memories is not as critical as for other types of memories such as SRAM and DRAM where the area overhead is mostly determined by the number of redundant bits. This allows for a flexible design of more powerful error correcting codes for NAND flash memories.

Similar to SLC flash memories, the primary failure mechanisms for MLC NAND flash memories include threshold voltage distribution, program/read disturb, data retention, programming/erasing endurance and single event upset. However, while for SLC flash memories a lot of errors are asymmetric, e.g. errors introduced by program disturb and data retention (Cassuto et al., 2007), for MLC NAND flash memories errors have no preferred symmetry (Yaakobi et al., 2009). Moreover, experimental results show that errors in MLC flash memories are more likely to occur uniformly within a page without any observable burstiness or local data dependency (Yaakobi et al., 2009). Thereby, throughout the paper we assume a random symmetric error model. Let c be the error-free output of the memory and e be the error vector. The distorted output $\tilde{c}$ can be written as $c \oplus e$, where $\oplus$ is the XOR operation. The probability of a non-zero error e can be computed as $P(e) = \theta^{||e||}(1-\theta)^{n-||e||}$, where θ is the raw bit distortion rate and $||e||$ is the multiplicity (the number of non-zero components) of the error. We want to emphasize that the proposed nonlinear multi-error correcting codes not only have advantages over linear codes under this error model but can also provide a guaranteed level of reliability in situations where the error model is unpredictable or multi-bit errors are more probable.

7.2.2 Error Correcting Algorithms for Nonlinear Multi-Error Correcting Codes

In Chapter 3, the constructions of two nonlinear multi-error correcting codes were shown. In this Section, we present the error correcting algorithms and analyze the error detecting and correcting properties of these two codes.

The Error Correcting Algorithm for Codes Based on Concatenation

The first construction of nonlinear multi-error correcting codes is based on the idea of concatenating linear and nonlinear redundant digits (see Theorem 3.4.7 in Chapter 3). Let $f : GF(p^k) \rightarrow GF(p^{r_n})$ be a nonlinear function with nonlinearity P_f. Let $V = \{(z, \phi(z))\}$ be a linear code with Hamming distance d, where $z \in GF(p^{k+r_n})$ and $\phi(z) : GF(p^{k+r_n}) \rightarrow GF(p^{r_l})$ is the encoding function. The codewords are in the format of $(y, f(y), \phi(z))$, where $y \in GF(p^k), f(y) \in GF(p^{r_n}), z = (y, f(y)) \in GF(p^{k+r_n})$ and $\phi(z) \in GF(p^{r_l})$.

Table 7.6: The output of the decoder for linear codes that can correct up to t errors

Cases	Error Vector	Error Flag Signal
No errors are detected	**0**	0
Errors of multiplicity at most t are detected	Correctable error vector	Multiplicity of the error
Errors of multiplicity larger than t are detected	**0**	-1

Let $e = (e_1, e_2, e_3)$ be the error vector, $c = (x_1, x_2, x_3)$ be the original codeword and $\tilde{c} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$ be the distorted codeword, where $e_1, x_1, \tilde{x}_1 \in GF(p^k)$, $e_2, x_2, \tilde{x}_2 \in GF(p^{r_n}), e_3, x_3, \tilde{x}_3 \in GF(p^{r_l})$ and $\tilde{x}_i = x_i \oplus e_i, 1 \leq i \leq 3$. Let $S = f(\tilde{x}_1) \oplus \tilde{x}_2$ be the nonlinear syndrome of the code that can be used for error detection. Assume a standard decoder for V is available. The output of the decoder is composed of the error flag signal E_V and the possible error vector $(\hat{e}_1, \hat{e}_2, \hat{e}_3)$. The values of E_V and the error vector in different situations are defined in Table 7.6.

The detailed error correcting algorithm for code C is described in Algorithm 1. The algorithm only corrects errors when information digits are distorted. If errors only occur in the redundant digits or errors are uncorrectable, no correction will be attempted.

Algorithm 1: Error correcting algorithm for the nonlinear multi-error correcting codes in Theorem 3.4.7

Input : $\tilde{c} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$
Output: $e = (e_1, e_2, e_3)$, ERR

```
1  begin
2      Decode V, compute S;
3      if E_V = 0, S = 0 then
4          No errors are detected, ERR = 0;
5      else if E_V = 0, S ≠ 0 then
6          Uncorrectable multi-errors are detected, ERR = 1;
7      else if E_V = −1 then
8          Uncorrectable multi-errors are detected, ERR = 1;
9      else
10         E_V > 0;
11         if ê_1 = 0 then
12             Errors in the redundant digits are detected, ERR = 0;
13         else
14             Compute x̂_1 = x̃_1 ⊕ ê_1, x̂_2 = x̃_2 ⊕ ê_2 ;
15             Compute Ŝ = f(x̂_1) ⊕ x̂_2;
16             if Ŝ = 0 then
17                 e = (ê_1, ê_2, ê_3), ERR = 0 ;
18             else
19                 Uncorrectable multi-errors are detected, ERR = 1
```

Theorem 7.2.1 *The nonlinear multi-error correcting codes presented in Theorem 3.4.7 have no errors miscorrected for all codewords. Any non-zero error will be miscorrected as an error* $e = (\hat{e}_1, \hat{e}_2, \hat{e}_3), ||e|| \leq t, \hat{e}_1 \neq 0$ *with a probability of at most* P_f.

Proof In Algorithm 1, an error e will be miscorrected if and only if $E_V > 0$, $\hat{e}_1 \neq 0$, $(\hat{e}_1, \hat{e}_2, \hat{e}_3) \neq (e_1, e_2, e_3)$ and $\hat{S} = 0$. When $E_V > 0$, after correcting possible errors in

$\tilde{x}_1$ and $\tilde{x}_2$, the effective error vector when computing $\hat{S}$ is $(e_1 \oplus \hat{e}_1, e_2 \oplus \hat{e}_2)$. Thereby $\hat{S} = f(x_1 \oplus e_1 \oplus \hat{e}_1) \oplus x_2 \oplus e_2 \oplus \hat{e}_2$. Since $x_2 = f(x_1)$, $\hat{S} = 0$ can be re-written as

$$f(x_1 \oplus e_1 \oplus \hat{e}_1) \oplus f(x_1) \oplus e_2 \oplus \hat{e}_2 = 0. \tag{7.9}$$

Let H be the parity check matrix of V. When (e_1, e_2, e_3) is miscorrected as $(\hat{e}_1, \hat{e}_2, \hat{e}_3)$, we have $H(\hat{e}_1, \hat{e}_2, \hat{e}_3) = H(e_1, e_2, e_3)$. Thereby $(\hat{e}_1, \hat{e}_2, \hat{e}_3) \oplus (e_1, e_2, e_3)$ is a codeword of V. If $(\hat{e}_1, \hat{e}_2, \hat{e}_3) \neq (e_1, e_2, e_3)$, then $(\hat{e}_1, \hat{e}_2)$ cannot be equal to (e_1, e_2). Otherwise to guarantee that $(\hat{e}_1, \hat{e}_2, \hat{e}_3) \oplus (e_1, e_2, e_3)$ is a codeword of V, $\hat{e}_3$ has to be equal to e_3, which contradicts to the assumption that $(\hat{e}_1, \hat{e}_2, \hat{e}_3) \neq (e_1, e_2, e_3)$. Thereby the miscorrected errors can be divided into two cases as stated below.

1. If $\hat{e}_1 \neq e_1$, from the definition of the nonlinear function, there are at most $P_f p^k$ solutions for x_1 in (7.9). Thereby, the error will be miscorrected with a probability of at most P_f.

2. If $\hat{e}_1 = e_2$, then $\hat{e}_2 \neq e_2$. The error will always be detected since (7.9) will never be satisfied.

The nonlinear multi-error correcting code presented in Theorem 3.4.7 has no undetectable errors and no errors miscorrected for all codewords. Its encoding and decoding area overhead is comparable to the linear code V (Section 7.2.5). When $p = 2^m$, the presented code has the same digit-error and burst-error detecting and correcting capabilities as V. We note that the proposed code has errors that are conditionally detected or miscorrected. However, most of these errors are of a high multiplicity and are less dangerous assuming that errors with small multiplicities are more probable. Moreover, these errors will be detected with a probability of at least $1 - P_f$, where P_f is the nonlinearity of f. Thereby, the reliability of architectures protected by the presented nonlinear multi-error correcting codes can be further im-

proved by increasing the nonlinearity for f. (To increase the nonlinearity of f, it may be necessary to increase the number of redundant digits for V (Carlet and Ding, 2004).)

While being able to provide an improved protection of systems comparing to linear error correcting codes, the nonlinear multi-error correcting codes generated by Theorem 3.4.7 still have the following shortages. First, it always requires more redundant digits than linear codes with the same error correcting capabilities. Second, the error correcting algorithm cannot be easily modified to correct errors beyond the error correcting capability t. Obviously, corrections of some errors with multiplicity larger than t can further improve the reliability of the system. Although in the literature there are works discussing the beyond-t error corrections for linear error correcting codes, the modified algorithm usually has much higher area or timing complexity, see e.g. (Sudan, 1997).

The Error Correcting Algorithm for Multi-Error Correcting Generalized Vasil'ev Codes

The second construction of nonlinear multi-error correcting codes is generalized from Vasil'ev constructions (Vasil'ev, 1962). The presented codes may have the same number of redundant digits as linear Hamming codes or BCH codes (see Theorem 3.4.8 in Chapter 3). Moreover, the error correcting algorithm for the presented codes can also be used to correct some errors with multiplicities larger than t requiring no extra area and timing overhead when comparing to schemes that only correct errors with multiplicities up to t.

Let $q_1 = p^{l_1}$ and $q_2 = p^{l_2}$, $l_1 \geq l_2 \geq 1$. Let V be a (n_1, k_1, d) q_1-ary code and $U = \{(u, uP)\}$ be a (n_2, k_2, d') q_2-ary code, where $u \in GF(q_2^{k_2}), k_2 \leq n_1$, $r_2 = n_2 - k_2$, $d' \geq d - 1$ and P is a $k_2 \times r_2$ encoding matrix in $GF(q_2)$ (the last r_2 columns of the generator matrix of the code in standard form). Let $f : GF(q_1^{k_1}) \rightarrow GF(q_2^{r_2})$ be an

arbitrary mapping such that $f(\mathbf{0}), \mathbf{0} \in GF(q_1^{k_1})$ is equal to zero and $f(y) \oplus f(y') \neq f(y \oplus y')$ for some $y, y' \in GF(q_1^{k_1})$, where $\oplus$ is the digit-wise addition in $GF(p)$. Let $u = (u_1, u_2, \cdots u_{k_2})$, where $u_i \in GF(q_2)$. Let $\beta(u) = ((u_1, \mathbf{0}), (u_2, \mathbf{0}), \cdots, (u_{k_2}, \mathbf{0}))$, where $\mathbf{0} \in GF(p^{l_1 - l_2}), (u_i, \mathbf{0}) \in GF(q_1)$ and $\beta(u) \in GF(q_1^{k_2})$. Denote by v_k the information bits of $v \in V$. The code is composed of all vectors $(u, (\beta(u), \mathbf{0}) \oplus v, uP \oplus f(v_k))$.

We next describe the error correcting algorithm for the nonlinear multi-error correcting codes presented in Theorem 3.4.8. We consider elements of U in $GF(q_2)$ and elements of V in $GF(q_1)$ as equivalent digits. The multiplicity of the error is defined as the number of non-zero digits in the error vector. Let $e = (e_1, e_2, e_3)$ be the error vector and $\tilde{c} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$ be the distorted codeword, where $e_1, \tilde{x}_1 \in GF(q_2^{k_2})$, $e_2, \tilde{x}_2 \in GF(q_1^{n_1}), e_3, \tilde{x}_3 \in GF(q_2^{r_2})$ and $\tilde{x}_i = x_i \oplus e_i, 1 \leq i \leq 3$. Denote by $\tilde{v} = (\beta(\tilde{x}_1), \mathbf{0}) \oplus \tilde{x}_2$ the distorted codeword in V and $\tilde{v}_k$ the information part of $\tilde{v}$. In the presented error correcting algorithm, we assume that $d' = d$ (see Theorem 3.4.8) and the standard error correcting algorithms are available for V and U. After receiving the possibly distorted codeword $(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$, compute $S = \tilde{x}_1 P \oplus f(\tilde{v}_k) \oplus \tilde{x}_3$. Decode $\tilde{v}$ using the standard error correcting algorithm for V. The output of the decoder for V contains two parts. One is the decoded error vector $\hat{e}_2$ in $GF(q_1^{n_1})$ and the other is the error flag signal E_V. Similarly, the outputs of the decoder for U contain the error flag signal E_U and the possible error patterns for the first and the third part of the codeword, which are $\hat{e}_1 \in GF(q_2^{k_2})$ and $\hat{e}_3 \in GF(q_2^{r_2})$ respectively. The values of the output signals of the decoders for U and V in different cases are described in Table 7.6. The detailed error correcting algorithm for the nonlinear multi-error correcting code presented in Theorem 3.4.8 is shown in Algorithm 2.

In reality, the error will be corrected only if at least one of the information bits is distorted. If all errors are in the redundant bits, no correction will be attempted. The

Algorithm 2: Error correcting algorithm for the nonlinear multi-error correcting codes in Theorem 3.4.8

Input : $\tilde{c} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$
Output: $e = (e_1, e_2, e_3)$, ERR

begin
  Decode V, compute $\hat{e}_2, E_V, S$;
  if $E_V = 0, S = 0$ **then**
    No errors are detected, $ERR = 0$;
  else if $E_V = 0, S \neq 0$ **then**
    if $d = 3$ **then**
      Errors either occur only in the redundant bits or are uncorrectable multi-errors;
      $ERR = 1$;
    else
      Decode U, compute $\hat{e}_1, \hat{e}_3, E_U$;
      if $E_U > 0$ **then**
        $e = (\hat{e}_1, (\beta(\hat{e}_1), \mathbf{0}), \hat{e}_3)$;
        e will only be corrected when $||e|| \leq t$;
        $ERR = 0$ when errors are corrected and $ERR = 1$ otherwise;
      else
        Uncorrectable muti-errors are detected, $ERR = 1$;
  else if $E_V = -1$ **then**
    Uncorrectable muti-errors are detected, $ERR = 1$;
  else
    E_V is larger than 0;
    Compute: ;
      $\hat{x}_2 = \tilde{x}_2 \oplus \hat{e}_2$;
      $\hat{v} = (\beta(\tilde{x}_1), \mathbf{0}) \oplus \hat{x}_2$;
      $\hat{x}_3 = \tilde{x}_3 \oplus f(\hat{v}_k)$ ($\hat{v}_k$ is the information part of $\hat{v}$);
    Decode U according to $\tilde{x}_1$ and $\hat{x}_3$;
    if $E_U = 0$ **then**
      $e = (0, \hat{e}_2, 0)$;
      e will be only corrected if there are errors in the information bits;
      $ERR = 0$ when errors are corrected and $ERR = 1$ otherwise;
    else if $E_U > 0$ **then**
      $e = (\hat{e}_1, (\beta(\hat{e}_1), \mathbf{0}) \oplus \hat{e}_2, \hat{e}_3)$;
      e will only be corrected when $||e|| \leq t$ and there are errors in the information bits;
      $ERR = 0$ when errors are corrected and $ERR = 1$ otherwise;
    else
      $E_U = -1$, uncorrectable multi-errors are detected, $ERR = 1$.

miscorrection of errors for C is strongly related to the error correcting properties of U and V. To simplify the analysis, we assume that U and V are both linear codes. For a given $e, 1 \leq ||e|| \leq t$, all vectors belonging to the coset in which e is the coset leader will be miscorrected as e by the linear code. For any $(n, k, d = 2t + 1)$ binary linear code that can correct up to t errors, for example, the number of miscorrected errors is $\sum_{i=1}^{t} \binom{n}{i}(2^k - 1)$. Compared to linear codes, the number of miscorrected errors for the proposed nonlinear multi-error correcting codes will be drastically reduced as shown in the following Theorem.

Theorem 7.2.2 *Assume $d' = d$ in Theorem 3.4.8 and f is a perfect nonlinear function from $GF(q_1^{k_1})$ to $GF(q_2^{r_2})$. Suppose only errors occurring to the information bits are corrected, denote by T the number of correctable error patterns by the nonlinear multi-error correcting codes presented in Theorem 3.4.8. The number of errors miscorrected for all codewords of the multi-error correcting codes presented in Theorem 3.4.8 is $|K_c| = (q_2^{k_2} - 1)T$.*

Proof According to Algorithm 2, the occurrence of miscorrections can be divided into the following cases.

1. When $d > 3$, $E_V = 0, S \neq 0$, $E_U > 0$, the error pattern is $e = (\hat{e}_1, (\beta(\hat{e}_1), 0), \hat{e}_3)$ which is generated by the decoder of U. Errors are miscorrected by C if and only if they are miscorrected by the linear code U. Since e will be corrected only if $\hat{e}_1 \neq 0$ and $||e|| \leq t$, the number of possible correctable error patterns is

$$\sum_{i=1}^{\lfloor \frac{t}{2} \rfloor} \sum_{j=0}^{t-2i} \binom{k_2}{i} \binom{r_2}{j} (q_2 - 1)^{i+j}. \tag{7.10}$$

   Thereby the number of miscorrected errors in this class is

$$(q_2^{k_2} - 1) \times \sum_{i=1}^{\lfloor \frac{t}{2} \rfloor} \sum_{j=0}^{t-2i} \binom{k_2}{i} \binom{r_2}{j} (q_2 - 1)^{i+j}. \tag{7.11}$$

$$\sum_{i=1}^{t}\sum_{j=0}^{t-i}\sum_{l=0}^{t-i-j}\binom{k_1}{i}\binom{r_1}{j}\binom{r_2}{l}(q_1-1)^{i+j}(q_2-1)^{l}. \tag{7.13}$$

$$(q_2^{k_2}-1)\times\sum_{i=1}^{t}\sum_{j=0}^{t-i}\sum_{l=0}^{t-i-j}\binom{k_1}{i}\binom{r_1}{j}\binom{r_2}{l}(q_1-1)^{i+j}(q_2-1)^{l}. \tag{7.14}$$

$$\sum_{i=1}^{t}\sum_{j=0}^{t-i}\sum_{l=0}^{t-i-j}\binom{k_2}{i}\binom{n_1}{j}\binom{r_2}{l}(q_1-1)^{j}(q_2-1)^{i+l}-\sum_{i=1}^{\lfloor t/2\rfloor}\sum_{j=0}^{t-2i}\binom{k_2}{i}\binom{r_2}{j}(q_2-1)^{i+j}. \tag{7.15}$$

$$(q_2^{k_2}-1)\times\Big(\sum_{i=1}^{t}\sum_{j=0}^{t-i}\sum_{l=0}^{t-i-j}\binom{k_2}{i}\binom{n_1}{j}\binom{r_2}{l}(q_1-1)^{j}(q_2-1)^{i+l} - \sum_{i=1}^{\lfloor t/2\rfloor}\sum_{j=0}^{t-2i}\binom{k_2}{i}\binom{r_2}{j}(q_2-1)^{i+j}\Big). \tag{7.16}$$

---

2. When $E_V > 0$,

$$\hat{x}_3 = \tilde{x}_3 \oplus f(\hat{v}_k) = x_1 P \oplus f(v_k) \oplus f(\hat{v}_k) \oplus e_3. \tag{7.12}$$

After correcting $\hat{e}_2$, the error pattern visible to U is $(e_1, f(v_k) \oplus f(\hat{v}_k) \oplus e_3)$.

(a) If $\hat{v}_k \neq v_k$ ($(e_1, \mathbf{0}) \oplus e_2$ is miscorrected by V), $f(v_k) \oplus f(\hat{v}_k) \oplus e_3$ may vary for different information bits. An error $(e_1, f(v_k) \oplus f(\hat{v}_k) \oplus e_3)$ will be miscorrected if and only if it belongs to the same coset as correctable errors (including the all-zero error vector) by U. Since errors are only corrected when $||e|| \leq t$, it is easy to verify that none of errors in this class will always be miscorrected.

(b) If $\hat{v}_k = v_k$ ($(e_1, \mathbf{0}) \oplus e_2$ is successfully corrected by V), the error pattern visible to U will be (e_1, e_3). Errors in this class are miscorrected if and only if (e_1, e_3) is miscorrected by U.

i. When $E_U \geq 0$ and $\hat{e}_1 = 0$, the error will be only corrected if $\hat{e}_2$ has

errors in the information bits and $||e|| \leq t$. The number of correctable error patterns is given by (7.13). The number of miscorrected errors in this situation is given by (7.14).

ii. When $E_U > 0$ and $\hat{e}_1 \neq 0$, the number of correctable error patterns is (7.15). The number of miscorrected errors in this case is given by (7.16).

Nonlinear multi-error correcting codes presented in Theorem 3.4.8 still have errors undetected or miscorrected for all codewords. However, the number of these errors are drastically reduced compared to linear codes with the same length and the same error correcting capability (see Section 7.2.3).

In Theorem 3.4.8, elements of U and V can belong to different fields, which allows a more flexible selection of the two codes. For example, in order to construct a nonlinear 5-digit error correcting code with elements in $GF(2^{10})$, we can select V to be a linear 5-digit error correcting Reed-Solomon code in $GF(2^{10})$ and U to be a binary repetition code with $n = 11$ and $k = 1$ with elements in $GF(2)$. The resulting code will have the same digit-error correcting capabilities, much less undetectable and miscorrected errors at the cost of only one more redundant digit in $GF(2^{10})$ (redundant bits of U) when comparing to V.

The codes presented in Theorem 3.4.8 may be as good as BCH codes in terms of the number of redundant digits (Example 3.4.6). Moreover, Algorithm 2 can be slightly modified to correct errors with multiplicities larger than t. For example, when $E_V = 0, S \neq 0$ and $E_U > 0$, the potential error pattern is $e = (\hat{e}_1, (\beta(\hat{e}_1), \mathbf{0}), \hat{e}_3)$. The original algorithm only correct errors when $||e|| \leq t$. This requirement can be removed so that some errors with multiplicity larger than t can also be corrected.

Example 7.2.1 *In this example we describe the encoding and decoding procedure of a* $(31, 17, 5)$ *binary nonlinear 2-error-correcting code. Let* V *be a* $(26, 16, 5)$ *BCH code*

whose generator polynomial is $g(x) = x^{10}+x^9+x^8+x^6+x^5+x^3+1$. *Let* $U = \{(u, uP)\}$ *be a repetition code, where* $u \in GF(2), uP \in GF(2^4)$. *Select* f *to be a quadratic perfect nonlinear function from* $GF(2^{16})$ *to* $GF(2^4)$ *defined by* $f = s_1 \bullet s_2 \oplus s_3 \bullet s_4$, *where* $\oplus$ *is the bit-wise XOR and* $\bullet$ *is the multiplication in* $GF(2^4)$. *Let* (10101100111101001) *be the* 17*-bit message that needs to be encoded. Then* $u = 1$, $v_k = (1101100111101001)$. *The redundant bits for* V *is* (0101110001) *and* $Pu \oplus f(v_k) = (1001)$. *Thereby the entire codeword is*

$$c = (10101100111101001010111000110 01).$$

Suppose the four left-most bits are distorted. The distorted codeword is

$$\tilde{c} = (01011100111101001010111000110 01).$$

Thereby we have

$$\tilde{v} = (10111001111010010101110001).$$

The decoder will correct the 2*-bit error* $\hat{e}_2 = (01100000000000000000000000)$ *in* $\tilde{v}$. *After this,* $\hat{v}$ *and* $\hat{x}_3$ *are re-computed and* U *is decoded according to* $\tilde{x}_1$ *and* $\hat{x}_3$. *It is easy to verify that* $\hat{x}_3 = (1111)$. *Since* $\tilde{x}_1 = 0$ *(the first bit of* $\tilde{c}$*), the input to the decoder of* U *is* (01111). *An error in the first bit will be successfully corrected by* U. *Hence we have*

$$\begin{aligned} e &= (\hat{e}_1, (\beta(\hat{e}_1), \mathbf{0}) \oplus \hat{e}_2, \hat{e}_3) \\ &= (11110000000000000000000000000). \end{aligned}$$

Thereby, an error of multiplicity four is successfully corrected although the Hamming distance of the code is only five.

We note that correcting errors with multiplicity larger than t will result in more errors that are miscorrected for all codewords. For example, when $E_V = 0, S = 0$ and $E_U > 0$, the error $e = (e_1, e_2, e_3)$ is miscorrected if and only if (e_1, e_3) is miscorrected by U. Suppose no correction will be attempted if $||e|| > t$, errors will not be corrected if $||\hat{e}_1|| + ||\hat{e}_3|| = t$ since in that case the resulting error vector $(\hat{e}_1, (\beta(\hat{e}_1), (0)), \hat{e}_3)$ will have a multiplicity larger than t. As a result, no miscorrections will happen when

$(\hat{e}_1, \hat{e}_3) \neq (e_1, e_3)$, $||\hat{e}_1|| + ||\hat{e}_3|| = t$. Without the limitation on $||e||$, miscorrections will occur when (e_1, e_3) with a multiplicity of $t+1$ is mistakenly corrected as $(\hat{e}_1, \hat{e}_3)$ whose multiplicity is t. Thereby, there is a tradeoff between the number of correctable error patterns and the size of the correction kernel $|K_C|$. The decision of whether and how to modify Algorithm 2 should be made according to specific applications and the estimated error models.

7.2.3 Alternatives for the Protection of MLC NAND Flash Memories

In this Section we compare six 5-bit error correcting codes for the protection of MLC NAND flash memories with 1024 data bytes in each page. For MLC NAND flash memories with a larger page size, longer codes generated by Theorem 3.4.7 and 3.4.8 can be used and all the analysis and comparison can be easily adjusted to justify the advantage of the presented codes.

The first two alternatives are the widely used $(8262, 8192, 11)$ BCH code (Sun et al., 2007) and the $(830, 820, 11)$ shortened Reed-Solomon code defined over $GF(2^{10})$ (Chen et al., 2008). The third and the forth alternatives are based on Theorem 3.4.7. Let $p = 2, k = 8200, r_n = 10$. Select f to be a quadratic perfect nonlinear function defined by the following equation.

$$f(y) = y_1 \bullet y_2 \oplus y_3 \bullet y_4 \cdots y_{819} \bullet y_{820}, \tag{7.17}$$

where $y_i \in GF(2^{10}), 1 \leq i \leq 820$ and $\bullet$ is the multiplication in $GF(2^{10})$. Let V be a $(8280, 8210, 11)$ BCH code. The codeword of the resulting nonlinear multi-bit correcting code constructed as described in Theorem 3.4.7 is in the format of $(y, f(y), \phi(z))$, where $z = (y, f(y))$, $y \in GF(2^{8200})$ are the information bits, $f(y) \in GF(2^{10})$ are the nonlinear redundant bits and $\phi(z) \in GF(2^{70})$ are the linear redundant bits. The code is a 5-bit error correcting code with length 8280 and dimension 8200.

Let $p = 2^{10}, k = 820$ and $r_n = 1$. Let f be the same quadratic function defined in (7.17) and V be a $(831, 821, 11)$ shortened Reed-Solomon codes in $GF(2^{10})$. A $(831, 820, 11)$ nonlinear multi-digit error correcting code can be constructed by Theorem 3.4.7. The code has the same bit-error correcting capabilities as the $(8262, 8192, 11)$ BCH codes. Moreover, the code can also correct burst errors and up to 5-digit errors like $(830, 820, 11)$ Reed-Solomon codes defined over $GF(2^{10})$.

The fifth and the sixth alternatives are based on Theorem 3.4.8. For these two alternatives, f is still selected to be the quadratic function defined in (7.17). Let $q_1 = q_2 = 2$. Let V be a $(8270, 8200, 11)$ BCH code and U be a $(11, 1, 11)$ linear repetition code. The nonlinear code constructed as described in Theorem 3.4.8 is a $(8281, 8200, 11)$ nonlinear 5-bit error correcting code. The code can also correct some errors with multiplicities higher than 5 without any extra overhead.

Let $q_1 = 2^{10}$ and $q_2 = 2$ in Theorem 3.4.8. Let V be a $(830, 820, 11)$ shortened Reed-Solomon code defined over $GF(2^{10})$ and U be a $(11, 1, 11)$ repetition code in binary field. The resulting code in Theorem 3.4.8 is a $(8301, 8201, 11)$ nonlinear 5-digit error correcting code. (The length and the dimension here are in terms of the number of binary bits.) The code can correct up to 5-digit errors in $GF(2^{10})$ and has the same burst error correcting capabilities as Reed-Solomon codes. (The first bit is treated as a separate digit.) Like alternative five, the nonlinear multi-digit error correcting code based on Theorem 3.4.8 can also correct some errors with more than 5 digits distorted.

Table 7.7 compares the error correcting properties of the six 5-bit error correcting codes. BCH code and Reed-Solomon code have a large number of undetectable errors. Let us denote the number of codewords of multiplicity i by A_i. For every codeword c of multiplicity eleven belonging to the BCH code, if $e \oplus e' = c, ||e|| = 5$ and $||e'|| = 6$, $||e'||$ will be miscorrected for all codewords as e since they have the same syndrome.

Table 7.7: Comparison of six 5-bit error correcting codes for the protection of MLC NAND flash memories

	BCH	Reed-Solomon[a]	Code 3	Code 4	Code 5	Code 6
$n^{(b)}$	8262	8300	8280	8310	8281	8301
$k^{(b)}$	8192	8200	8200	8200	8201	8201
Digit & Burst Error Correcting	No	Yes	No	Yes	No	Yes
Beyond 5-bit Error Correcting	No	No	No	No	Yes	Yes
Number of Undetectable Errors [e]	2^{8192}	2^{8200}	0	0	1	1
Miscorrected Errors [e], $\|\|e\|\| = 6$	$\approx 10^{17}$	$< A^*_{11}\binom{11}{6}$ [c]	0	0	0[d]	0[d]
Fraction of Miscorrected Errors	$\approx 10^{-4}$	$\approx 10^{-3}$	0	0	≈ 0	≈ 0

(a): The code is defined in $GF(2^{10})$.
(b): The length and the dimension here are in terms of the number of bits in binary field.
(c): A^*_{11} is the number of codewords in a code over $GF(2^{10})$ with 11 non-zero digits.
(d): The number is for the case when no errors with multiplicity larger than 5 are corrected.
(e): Errors undetectable or miscorrected for all codewords. Besides these two types of errors, the presented codes also have errors that are conditionally detectable or conditionally miscorrected, most of which can be detected with a probability of $1 - 2^{-10}$.

Thereby, the number of errors of multiplicity six miscorrected for all codewords of the BCH code is $A_{11}\binom{11}{6}$. (If the error is only corrected when there is at least one information bit distorted, this number will be a little smaller.) According to the results presented in (MacWilliams and Sloane, 1998), for a $(8262, 8192, 11)$ shortened BCH code, A_{11} can be roughly estimated by $A_{11} = \binom{8262}{11}/2^{70} \approx 10^{14}$. Thereby, a large number of errors with multiplicity 6 will be miscorrected by the BCH codes. The fraction F of errors that are always miscorrected by the BCH code can be calculated as

$$F = \frac{(2^k - 1)\sum_{i=1}^{t}\binom{n}{i}}{2^n} \approx \frac{\sum_{i=1}^{t}\binom{n}{i}}{2^{n-k}}. \tag{7.18}$$

For the linear $(8262, 8192, 11)$ BCH code, $F \approx 10^{-4}$.

For the Reed-Solomon code, a 6-bit error is miscorrected if and only if (1) the 6 bits are spread over 6 digits in $GF(2^{10})$; and (2) there is a codeword with 11 digits that contains the 6 digits in (1). Moreover, given the fact that the length of the Reed-Solomon code (in terms of digits) is much smaller than the length of the BCH code (in terms of bits), the number of miscorrected 6-bit errors for Reed-Solomon codes will be much smaller than that for BCH codes. However, similar to (7.18), we can

compute the proportion of errors that are always miscorrected by the Reed-Solomon code, which is of the order of 10^{-3} and is worse than the BCH code.

Codes 3 and 4 generated by Theorem 3.4.7 do not have undetectable and miscorrected errors. Codes 5 and 6 based on Theorem 3.4.8 have only 1 undetectable error. When no error with multiplicity larger than 5 is corrected, according to the proof of Theorem 3.4.8 it is easy to derive that the smallest possible multiplicity of errors that are always miscorrected by code 5 or 6 is seven. Moreover, since $q_2 = 2, k_2 = 1$, the number of errors that are miscorrected for all codewords is equal to the number of correctable errors. As a result, the fraction of errors that are always miscorrected by these two codes is almost 0. When each non-zero error pattern is equi-probable, the probability of miscorrection for the presented nonlinear multi-error correcting codes is much smaller than that for BCH codes and Reed-Solomon codes. Moreover, the presented codes have no errors of multiplicity 6 that are miscorrected for all codewords. Thereby, when assuming a fixed bit error rate, the probability that an error is always miscorrected by these codes is still much smaller than their linear alternatives.

Different from linear codes, the four presented nonlinear multi-error correcting codes have errors that are conditionally undetectable or miscorrected. These errors usually have high multiplicities. Moreover, most of these errors can be detected with a probability $1 - 2^{-10}$. Thereby, the existence of conditionally undetectable or miscorrected errors will not compromise the reliability of NAND flash memories protected by the proposed nonlinear multi-error correcting codes.

We note that the above message-dependent error detecting characteristic of the proposed nonlinear multi-error correcting code is also very helpful for MLC NAND flash memories for the protection of hardware malfunctions such as data retention and programming/erasing endurance failure (Bez et al., 2003). Due to the decreased programming voltage margin, data retention is more likely to happen for MLC tech-

nology than for SLC technology. The problem of programming/erasing endurance also becomes more serious for MLC NAND flash memories, for which the typical number of supported program/erase cycles is fewer than 10000 (Cooke, 2007). Errors introduced by these hardware failures will never disappear or will only disappear after the next erasing or programming operation. Hence, the proposed nonlinear t-error-correcting code with stronger error detecting and correcting capability for repeating errors due to the message-dependent error detecting characteristic can be used together with other protection schemes to efficiently detect these failures and protect the devices against them.

Generally speaking, the proposed nonlinear multi-error correcting codes require more redundant bits than their linear alternatives. However, this results in only a very small decrease in the code rate (10 more redundant bits are required for more more than 8000 information bits). Moreover, due to the existence of spare bytes in MLC NAND flash memory (Section 7.2.1), the number of redundant bits of the error correcting codes used for MLC NAND flash memories is not as critical as for other types of memories such as SRAM and DRAM where the area overhead is mostly determined by the number of redundant bits.

7.2.4 Hardware Design of the Encoder and the Decoder for Nonlinear Multi-Error Correcting Codes

In this section, we present the encoder and the decoder architectures for the proposed nonlinear multi-error correcting codes. We estimate the area, the performance and the power consumption of the proposed architectures and compare them to architectures based on linear BCH codes and linear Reed-Solomon codes (see Section 7.2.3).

Encoder Architecture

The encoder for BCH codes and Reed-Solomon codes are conventionally implemented based on a linear feedback shift register (LFSR) architecture. Both the serial and the parallel structures for LFSRs are well studied in the community. In general, the serial LFSR needs k clock cycles while the parallel LFSR needs only $\lceil k/q \rceil$ clock cycles to finish the computation of the redundant bits at the cost of higher hardware complexity, where k is the number of information bits and q is the parallelism level of the LFSRs.

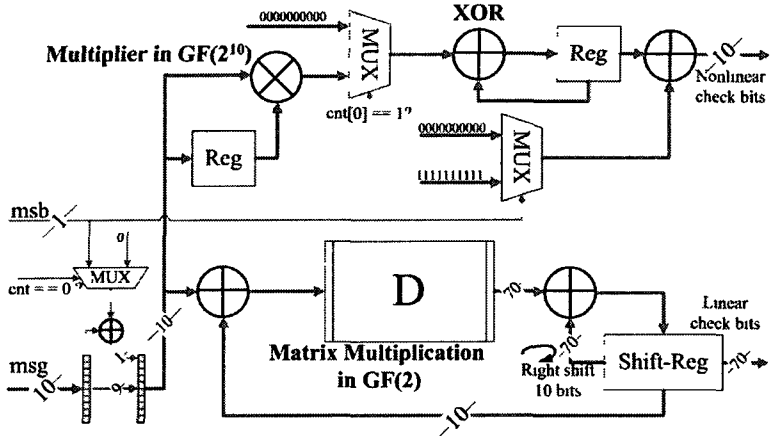


Figure 7·6: The architecture of the encoder for the $(8281, 8201, 11)$ nonlinear 5-error-correcting code

Compared to the encoder for the BCH codes and Reed-Solomon codes, the encoder for the proposed nonlinear multi-error correcting codes requires one more finite field multiplier and two registers for the computation of the nonlinear redundant bits. The detailed architecture of the encoder for the nonlinear $(8281, 8201, 11)$ 5-bit error correcting code generated by Theorem 3.4.8 is shown in Figure 7·6. The design is based on the parallel LFSR proposed in (Pei and Zukowski, 1992). The parallelism level of the design is 10. During each clock cycle, 10 information bits are inputted to the encoder. The most significant bit (msb) of the message is input via a separate port. The first information bit for the BCH code is derived by XORing msb with the

first bit of *msg* at the first clock cycle (when $cnt = 0$ as shown in the figure). The bottom half of the architecture is a parallel LFSR used to generate the redundant bits for BCH codes. D is a 10×70 binary matrix (Pei and Zukowski, 1992). During each clock cycle, the 10 most significant bits in the shift register are XORed with the new input and then multiplied by D. The output of the multiplier is XORed with the shifted data from the shift register to generate the input to the register. The top half of the architecture is for the computation of nonlinear redundant bits. During the even-numbered clock cycles, the 10-bit input is buffered. During the odd-numbered clock cycles, the buffered data is multiplied by the new input in $GF(2^{10})$ and then added to the output registers. A 10-bit mask is XORed with the data in the output register to generate the nonlinear redundant bits. For the $(8281, 8201, 11)$ 5-error-correcting code, 820 clock cycles are required to complete the encoding of the message.

The encoder for the $(8280, 8200, 11)$ nonlinear 5-bit error correcting code based on Theorem 3.4.7 is similar to the one shown in Figure 7·6. The same structure (top half) is used to compute the 10-bit nonlinear redundant bits. The main difference between the two encoders is as follows. First, the encoder for the $(8280, 8200, 11)$ code does not require a separate port for *msb*. All information bits are input via *msg* in 820 clock cycles, assuming a parallelism level of 10. Second, the encoding of the $(8280, 8200, 11)$ code needs one more clock cycle to complete compared to the $(8281, 8201, 11)$ code. At the 821^{th} clock cycle, the input to D (Figure 7·6) is switched to the already-generated nonlinear check bits using a 10-bit 2:1 multiplexer.

Instead of using a parallel architecture described above, the encoders for the linear shortened $(830, 820, 11)$ Reed-Solomon code defined over $GF(2^{10})$ and the two nonlinear multi-digit error correcting codes presented in Section 7.2.3 can be based on a much simpler serial LFSR. Since the length of the Reed-Solomon code and the non-

linear multi-digit error correcting codes is 10 times shorter than that of the bit-error correcting codes, the number of clock cycles required to complete the encoding for the Reed-Solomon code and the nonlinear error correcting codes will still be the same as for the bit-error correcting codes even with a serial LFSR. The former, however, requires that all operations are performed in $GF(2^{10})$.

Decoder Architecture

The decoding of the proposed nonlinear multi-error correcting codes requires the decoding of a BCH code or a Reed-Solomon code. The standard decoder for the BCH codes mainly contains three parts: the syndrome computation block, the error locator polynomial generation block and the Chien search block (Cho and Sung, 2008). Compared to the decoder for the BCH codes, the decoder for the Reed-Solomon codes requires one more block to compute the error magnitude. We next briefly discuss the implementation of the above four blocks and then present the decoder architecture for the proposed nonlinear multi-error correcting codes.

1. **Syndrome Computation:** Without loss of generality, assume that the BCH code is a narrow-sense BCH code (MacWilliams and Sloane, 1998). Let us denote the received codeword by $\tilde{c} = (\tilde{x}_1, \tilde{x}_2 \cdots \tilde{x}_{n-1}, \tilde{x}_n)$. For a $(n, k, d = 2t+1)$ t-error-correcting BCH codes, the syndromes are defined as $S_i = \sum_{j=0}^{n-1} x_{j+1}\alpha^{ij}, 0 \leq i \leq 2t-1$, where α is the primitive element of $GF(2^m)$. For binary BCH codes, $S_{2i} = S_i^2$. Hence only odd-numbered S_i needs to be computed from $\tilde{c}$. The other syndromes can be computed using a much simpler square circuit in $GF(2^m)$. To improve the throughput of the decoder, a parallel design can be applied to process multiple bits per clock cycle. Figure 7·7 shows the syndrome computation circuit with a parallelism level of q for one S_i. For the whole syndrome computation block, t such structures are needed.

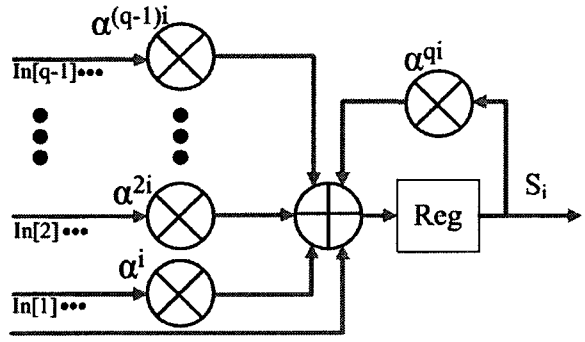


Figure 7·7: The syndrome computation block with a parallelism level of q for BCH codes

2. **Error Locator Polynomial Generation:** After the syndromes are computed, the error locator polynomial Λ will be generated using the Berlekamp-Massey (BM) algorithm. The hardware implementations of the BM algorithms have been well studied in the community (Burton, 1971; Sarwate and Shanbhag, 2001; Seth et al., 2006; Rizwan, 2008). In our design a fully serial structure proposed in (Burton, 1971) is used to minimize the area overhead. The design mainly requires three multipliers in $GF(2^m)$ and two FIFOs. The error locator polynomial Λ of degree t can be generated in $t(t+3)/2$ clock cycles. For our design, $t = 5$ and 20 clock cycles are needed for the generation of Λ.

3. **Chien Search:** Let us denote the primitive element in $GF(2^m)$ by α. The Chien search algorithm exhaustively tests whether α^i is a root of the error locator polynomial Λ. If $\Lambda(\alpha^i) = 0$, the error location is $2^m - 1 - i$. Rewrite $\Lambda(\alpha^i)$ as:

$$\begin{aligned} \Lambda(\alpha^i) &= \lambda_0 \oplus \lambda_1\alpha^i \oplus \lambda_2\alpha^{2i} \oplus \cdots \oplus \lambda_t\alpha^{ti} \\ &= \lambda_{0,i} \oplus \lambda_{1,i}\alpha \oplus \lambda_{2,i}\alpha^2 \oplus \cdots \oplus \lambda_{t,i}\alpha^t. \end{aligned} \tag{7.19}$$

The computation complexity is reduced based on the fact that $\lambda_{j,i+1} = \lambda_{j,i}\alpha^j, 0 \leq j \leq t$. The algorithm can also be paralleled to test multiple positions per clock

cycle. A typical implementation of the algorithm with a parallelism level of q contains t m-bit multiplexers and registers, $q \times t$ multipliers for multiplication by a constant and q adders in $GF(2^m)$ (Chen and Parhi, 2004). In (Cho and Sung, 2008), a strength-reduced parallel Chien search architecture is proposed. The authors showed that by a simple transformation of the error locator polynomial, most of the Galois field multiplications can be replaced by shift operations resulting in much lower hardware complexity (Figure 7·8). For the detail of the architecture, please refer to (Cho and Sung, 2008).

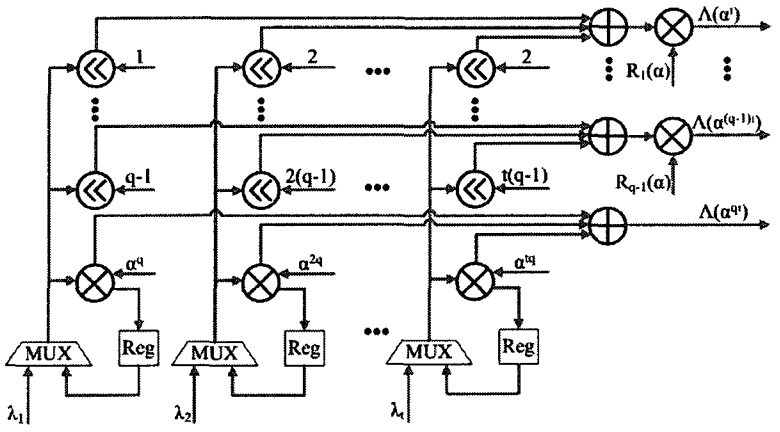


Figure 7·8: Strength-reduced Chien search architecture with a parallelism level of q

4. **Error Magnitude Computation for Reed-Solomon Codes:** Besides the error locator polynomial Λ, the Berlekamp-Massey algorithm can also generate the error magnitude polynomial $\Omega(z)$ defined by

$$(1 + S(z))\Lambda(z) = \Omega(z) \, mod \, z^{2t}, \tag{7.20}$$

where $S(z) = S_0 \oplus S_1 z \oplus \cdots \oplus S_{2t-1} z^{2t-1}$ is the syndrome polynomial. According to Forney's algorithm (Sarwate and Shanbhag, 2001), the error magnitude at

position i can be computed as

$$e_i = \frac{z^b \Omega(z)}{z\Lambda'(z)}, z = \alpha^{-i}, \tag{7.21}$$

where $\Lambda'(z)$ is the derivative of $\Lambda(z)$ and b is an integer. It is easy to verify that $\Lambda'(z)$ is simply the sum of the terms with odd degrees in $\Lambda(z)$ and can be directly derived during the computation of Λ.

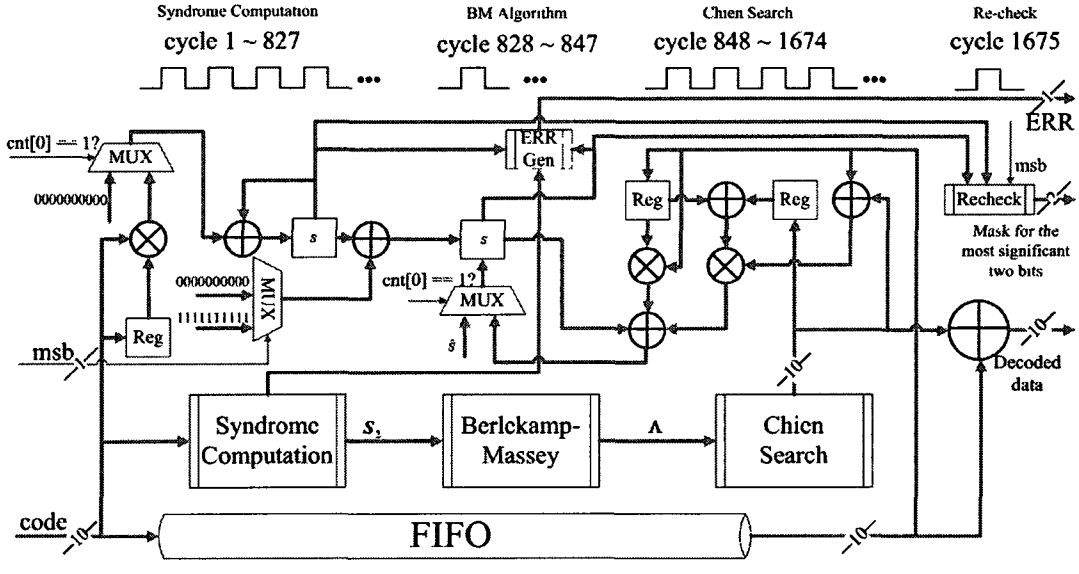


Figure 7.9: Decoder architecture for the proposed $(8281, 8201, 11)$ nonlinear 5-error-correcting code

The decoder for the nonlinear multi-error correcting codes presented in Theorem 3.4.7 is similar to the decoders for BCH codes and Reed-Solomon codes. In fact, most of the decoding can be completed by the standard BCH or Reed-Solomon decoder. The main difference is as follows. First, the nonlinear multi-error correcting codes need to compute the nonlinear syndrome S (see Algorithm 1) when receiving the possibly distorted codewords and re-compute S after correcting errors located by V. Second, after the decoding of the linear codes is completed and S is re-computed, one more clock cycle is required for the decoder of the nonlinear code to verify the error

correcting results so that possible miscorrection of errors can be prevented.

The decoder for the nonlinear multi-error correcting codes based on Theorem 3.4.8 is slightly more complicated than the decoder for codes based on Theorem 3.4.7. As an example, the detailed architecture of the decoder for the $(8281, 8201, 11)$ nonlinear 5-bit error correcting code is shown in Figure 7·9. The whole decoding procedure requires 1675 clock cycles assuming a parallelism level of 10. During the first 827 cycles, S and the syndrome of the BCH code are computed. If no errors are detected by the BCH code, the decoding procedure will be completed at the 828^{th} clock cycle. Depending on the value of S, either the first two information bits will be flipped or ERR will be pulled down by the ERR generating circuit which indicates that there are no errors occurring to the information bits of the code. The error locator polynomial generation and the Chien search will be incurred only when errors are detected by the BCH code, which can effectively reduce the average decoding latency.

If errors are detected by the BCH code, the Berlekamp-Massey algorithm will take another 20 clock cycles to generate the error locator polynomial Λ. After this the Chien search block will exhaustively test all possible error locations. If $\Lambda(\alpha^i) = 0$, then the error location is $2^m - 1 - i$. Since a $(8270, 8200, 11)$ shortened BCH code is used, only $\Lambda(\alpha^i), 8114 \leq i \leq 16383$ $(m = 14)$ need to be computed. The original strength-reduced Chien search architecture is slightly modified for the decoding of shortened BCH codes. The constant inputs $\lambda_i, 1 \leq i \leq t$ to the bottom t Galois field multipliers in Figure 7·8 are set to be α^{-10i} instead of α^{10i} $(q = 10)$.

$\hat{S}$ is initialized to be S and is serially updated during the Chien search stage. Starting from the 848^{th} clock cycle, the 10-bit FIFO output x_i (possibly distorted codeword) and the decoded 10-bit error vector e_i will be buffered in two 10-bit registers. At each odd-numbered clock cycle, $\hat{S}$ is updated as follows.

$$\hat{S} = \hat{S} \oplus x_{i-1} \bullet x_i \oplus (x_{i-1} \oplus e_{i-1}) \bullet (x_i \oplus e_i). \tag{7.22}$$

At the 1675 clock cycle, msb and $\hat{S}$ are used to re-check whether the most significant two bits are successfully corrected. A 2-bit error mask will be generated to make adjustment to these two bits according to the check results.

The decoder for the digit-error correcting code based on Theorem 3.4.8 presented in Section 7.2.3 and the decoder for the $(8281, 8201, 11)$ nonlinear 5-bit error correcting code are different as follows.

1. All operations of the decoder for the 5-digit error correcting code are performed in $GF(2^{10})$.

2. The 5-digit error correcting code does not require a parallel architecture. A serial design can achieve a similar decoding latency in terms of the number of clock cycles to the decoder for the $(8281, 8201, 11)$ 5-bit error correcting code with a parallelism level of 10.

3. One more block for the computation of the error magnitude is integrated into the architecture shown in Figure 7·9. The block is connected to the Chien search block and generates the final decoded memory contents.

The error magnitude polynomial is generated by the Berlekamp-Massey block. To reduce the hardware overhead, multipliers in $GF(2^{10})$ for the calculation of the nonlinear syndrome S are re-used to generate the error magnitude polynomial. One inverter in $GF(2^{10})$ is required to compute e_i according to Forney's algorithm (see (7.21)). In general, inverters in Galois field have much longer critical path than multipliers. Thus a 4-stage pipeline is added to reduce the latency of the inverter. Let $\sigma \in GF(2^{10})$, σ^{-1} can be represented as

$$\sigma^{-1} = \sigma^{2^{10}-2} = \sigma^{2^1+2^2+\cdots+2^9} \tag{7.23}$$

Given the fact that a 4-stage pipeline is implemented, the above function can be realized using square operations and 5 multiplications in $GF(2^{10})$. Again we re-use the multipliers in other blocks for the purpose of reducing the hardware overhead. Since the square operation is simple in $GF(2^{10})$, the inverter adds minimal area overhead and has a latency similar to the Galois filed multiplier in our design.

7.2.5 Area, Latency and Power Consumption

The area, latency and the power consumption for architectures based on the six alternatives presented in Section 7.2.3 are shown in Table 7.8. The designs are modelled in Verilog and synthesized in RTL Design Compiler using 45nm NANGATE library (Nangate, 2011). In practice the logic circuits used in NAND flash memory could be different from those used in standard digital designs. The estimation presented here is only for the purpose of investigating the increase in area, power and latency of architectures based on the proposed nonlinear multi-error correcting codes compared to architectures based on the widely used BCH codes and Reed-Solomon codes.

During the synthesis we fixed the clock rate for the encoder and the decoder and compared the area and the power consumption for architectures based on different codes. The encoders work at $1GHz$. The decoders work at a lower frequency – $400MHz$ – due to the long critical path in Berlekamp-Massey block (Chen et al., 2008). The six alternatives require the similar latency in terms of the number of clock cycles for encoding and decoding. Due to the computation of the error magnitude and the pipeline for the inverter in the Galois field, digit-error correcting codes (Reed-Solomon, etc) need 8 more clock cycles to complete the decoding compared to bit-error correcting codes (BCH, etc).

The encoders for the digit-error correcting codes require $40\% \backsim 50\%$ more area overhead and power than the encoders for bit-error correcting codes due to the fact

that all operations are in $GF(2^{10})$. The decoders for digit-error correcting codes, however, require 20% $\backsim$ 30% less overhead in area and power because of a much simpler serial architecture.

Compared to BCH codes and Reed-Solomon codes, the proposed nonlinear multi-error correcting codes need about 10% $\backsim$ 20% more area and power in total for the encoder and the decoder and have the similar latency in terms of the number of clock cycles required to complete the encoding and decoding. The $(8281, 8201, 11)$ nonlinear 5-bit error correcting codes based on Theorem 3.4.8 (column 6 and 7 in Table 7.8), for example, requires 17.5% more area and consumes 10.0% more power in total for the encoder and the decoder compared to the $(8262, 8192, 11)$ BCH code.

We note that the encoder and the decoder are only a very small portion in the MLC NAND flash memory chip, where the major portion is the memory cell array. Thereby the increase in area overhead for the encoder and the decoder is not significant for the reliable memory design. The power for ECC schemes is mostly consumed by the decoder. However, when there are no errors, which is the most probable case, the only active part in the decoder is the syndrome computation block. Thereby, in practice the average increase of the power consumption for the presented nonlinear multi-error correcting codes will be smaller than the data shown in Table 7.8. Given the fact that the reliability of MLC NAND flash memories protected by the proposed nonlinear multi-error correcting codes are much higher than those protected by linear codes (Table 7.7), the small increase in area and power is reasonable and acceptable.

7.3 Summary

We propose to use nonlinear error correcting codes to improve the reliability of memory systems. Nonlinear SEC-DED codes generalized from Vasil'ev codes and Phelps codes can be used for the build of reliable SRAMs tolerant to SEUs, etc. Nonlinear

multi-error correcting codes based on concatenations and Vasil'ev constructions can be used to protect MLC NAND flash memories, where multi-bit (or multi-digit) error correcting is essential for the stability and reliability of the device.

Architectures based on nonlinear error correcting codes have less undetectable errors and less errors that are misscorrected for all codewords than architectures based on linear codes. The error correcting algorithms for the proposed nonlinear codes are described. The multi-error correcting code generalized from Vasil'ev codes can also correct some errors with multiplicity larger than its error correcting capability t without any extra overhead in area, latency and power consumption compared to schemes that correct only up to t errors.

The encoders and decoders for different codes are modelled in Verilog and synthesized in RTL design compiler. The results show that architectures based on nonlinear error correcting codes can have nearly no undetectable and miscorrected errors while consuming about $15\% \sim 30\%$ more area and power consumption compared to architectures based on the linear codes with the same error correcting capability.

Table 7.8: Comparison of the area, the latency and the power consumption of different alternatives that can correct up to 5-bit errors for the protection of MLC NAND flash memories

	Bit-Error Correcting						Digit-Error Correcting					
	BCH		Theorem 3.4.7		Theorem 3.4.8		RS		Theorem 3.4.7		Theorem 3.4.8	
	ENC	DEC	ENC	DEC	ENC	DEC	ENC	DEC	ENC	DEC	ENC	DEC
Parallelism Level	10	10	10	10	10	10	1	1	1	1	1	1
Speed(Hz)	1G	400M	1G	400M	1G	400M	1G	400M	1G	400M	1G	400M
Latency(Cycles)	820	1674	821	1675	820	1675	820	1682	821	1683	820	1683
Area(μm^2)	854.4	12501.7	1468.3	14114.0	1459.8	14245.9	1407.4	9597.8	2051.7	11059.2	2094.5	11127.3
Power(mW)	0.61	3.31	0.85	3.37	0.82	3.48	0.91	2.03	1.13	2.14	1.20	2.18

References

Akdemir, K. D., Hammouri, G., and Sunar, B. (2009). Non-linear error detection for finite state machines. In *Proceedings of the 10th International Workshop on Information Security Applications (WISA)*, pages 226–238.

Akdemir, K. D., Wang, Z., Karpovsky, M. G., and Sunar, B. (book chapter to appear in 2011). *Design of Cryptographic Devices Resilient to Fault Injection Attacks Using Nonlinear Robust Codes.* in Fault Analysis in Cryptography.

Assmus, E. F., Jr, and Key, J. D. (1995). Polynomial codes and finite geometries, Manuscript. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.1139s`.

Atwood, G., Fazio, A., Mills, D., and Reaves, B. (1997). Intel StrataFlash™ memory technology overview. *Intel Technology Journal*, 1(2).

Bagatin, M., Cellere, G., Gerardin, S., Paccagnella, A., Visconti, A., Beltrami, S., and Maccarrone, M. (2007). Single event effects in 1Gbit 90nm NAND flash memories under operating conditions. In *13th IEEE International On-Line Testing Symposium, 2007. IOLTS 07.*, pages 146–151.

Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., and Whelan, C. (2006). The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382.

Baranov, S., Levin, I., Keren, O., and Karpovsky, M. (2009). Designing fault tolerant FSM by nano-PLA. *IEEE International On-Line Testing Symposium*, pages 229–234.

Barenghi, A., Bertoni, G., Parrinello, E., and Pelosi, G. (2009). Low voltage fault attacks on the RSA cryptosystem. *Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 23–31.

Berger, J. M. (1961). A note on an error detection code for asymmetric channels. *Information and Control*, 4:68–73.

Bertoni, G., Breveglieri, L., Koren, I., Maistri, P., and Piuri, V. (2003). Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. *IEEE Transactions on Computers*, 52(4):492–505.

Beth, T., Jungnickel, D., and Lenz, H. (1999). *Design Theory*, volume 1. Cambridge University Press.

Bez, R., Camerlenghi, E., Modelli, A., and Visconti, A. (2003). Introduction to flash memory. *Proceedings of the IEEE*, 91(4):489–502.

Bhattacharryya, D. K. and Nandi, S. (1997). An efficient class of SEC-DED-AUED codes. In *Proceedings of the 1997 International Symposium on Parallel Architectures, Algorithms and Networks*, ISPAN '97, pages 410–, Washington, DC, USA. IEEE Computer Society.

Boneh, D., Demillo, R. A., and Lipton, R. J. (2001). On the importance of eliminating errors in cryptographic computations. In *Journal of Cryptology*, volume 14, pages 101–119.

Bose, B. (1984). Unidirectional error correction/detection for VLSI memory. In *Proceedings of the 11th annual international symposium on computer architecture*, pages 242–244.

Bousselam, K., Natale, G. D., Flottes, M.-L., and Rouzeyre, B. (2010). Evaluation of concurrent error detection techniques on the advanced encryption standard. *IEEE International On-Line Testing Symposium*, pages 223–228.

Burton, H. (1971). Inversionless decoding of binary BCH codes. *IEEE Transactions on Information Theory*, 17(4):464–466.

Canivet, G., Maistri, P., Leveugle, R., Cldire, J., Valette, F., and Renaudin, M. (2010). Glitch and laser fault attacks onto a secure AES implementation on a SRAM-based FPGA. *Journal of Cryptology*, pages 1–22. 10.1007/s00145-010-9083-9.

Carlet, C. and Ding, C. (2004). Highly nonlinear mappings. *Journal of Complexity*, 20(2-3).

Cassuto, Y., Schwartz, M., Bohossian, V., and Bruck, J. (2007). Codes for multilevel flash memories: Correcting asymmetric limited-magnitude errors. In *IEEE International Symposium on Information Theory, 2007. ISIT 2007.*, pages 1176–1180.

Cellere, G., Gerardin, S., Bagatin, M., Paccagnella, A., Visconti, A., Bonanomi, M., Beltrami, S., Harboe-Sorensen, R., Virtanen, A., and Roche, P. (2009). Can atmospheric neutrons induce soft errors in NAND floating gate memories? *IEEE Electron Device Letters*, 30(2):178–180.

Chen, B., Zhang, X., and Wang, Z. (2008). Error correction for multi-level NAND flash memory using Reed-Solomon codes. In *IEEE Workshop on Signal Processing Systems, 2008. SiPS 2008.*, pages 94–99.

Chen, C. L. (1983). Error-correcting codes with byte error-detection capability. *IEEE Transactions on Computers*, C-32:615–621.

Chen, C. L. (1996). Symbol error correcting codes for memory applications. In *Proceedings of the The Twenty-Sixth Annual International Symposium on Fault-Tolerant Computing (FTCS '96)*, pages 200 –207.

Chen, T.-H., Hsiao, Y.-Y., Hsing, Y.-T., and Wu, C.-W. (2009). An adaptive-rate error correction scheme for NAND flash memory. In *27th IEEE VLSI Test Symposium, VTS '09.*, pages 53–58.

Chen, Y. and Parhi, K. (2004). Small area parallel Chien search architectures for long BCH codes. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(5):545–549.

Cho, J. and Sung, W. (2008). Strength-reduced parallel Chien search architecture for strong BCH codes. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(5):427–431.

Cooke, J. (2007). The inconvenient truths about NAND flash memory. Micron MEMCON'07 presentation. `http://download.micron.com/pdf/presentations/events/flash_mem_summit_jcooke_inconvenient_truths_nand.pdf`.

Cramer, R., Dodis, Y., Fehr, S., Padró, C., and Wichs, D. (2008). Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, pages 471–488, Berlin, Heidelberg. Springer-Verlag.

Dan, R. and Singer, R. (2003). White paper: Implementing MLC NAND flash for cost-effective, high capacity memory. M-Systems. `http://support.gateway.com/s/Manuals/Desktops/5502664`.

Degtyaryov, A. and Slyozkin, V. (2001). Conception of a generalized receiver in telecommunication systems. In *11th International Conference on Microwave and Telecommunication Technology, 2001. CriMiCo 2001*, pages 290–291.

Dodis, Y., Katz, J., and Reyzin, L. (2006). Robust fuzzy extractors and authenticated key agreement from close secrets. In *Advances in Cryptology - CRYPTO 06*, pages 232–250. Springer.

Dunning, L. A. (1985). SEC-BED-DED codes for error control in byte-organized memory systems. *IEEE Transactions on Computers*, 34:557–562.

Dutta, A. and Touba, N. A. (2007). Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code. In *25th IEEE VLSI Test Symposium (VTS'07)*, pages 349–354.

Eto, A., Hidaka, M., Okuyama, Y., Kimura, K., and Hosono, M. (1998). Impact of neutron flux on soft errors in mos memories. In *International Electron Devices Meeting, 1998. IEDM '98 Technical Digest.*, pages 367 –370.

Etzion, T. and Vardy, A. (1994). Perfect binary codes: Constructions, properties, and enumeration. In *IEEE Transaction on Information Theory*, volume 40, pages 754–763.

Gao, S. (1993). *Normal Bases over Finite Fields.* PhD thesis, University of Waterloo. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.67.2965`.

Gaubatz, G. and Sunar, B. (2006). Robust finite field arithmetic for fault-tolerant public-key cryptography. In *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 196–210.

Gaubatz, G., Sunar, B., and Karpovsky, M. G. (2006). Non-linear residue codes for robust public-key arithmetic. In *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC).*

Georgakos, G., Huber, P., Ostermayr, M., Amirante, E., and Ruckerbauer, F. (2007). Investigation of increased multi-bit failure rate due to neutron induced seu in advanced embedded srams. In *IEEE Symposium on VLSI Circuits, 2007*, pages 80 –81.

Halfhill, T. R. (2005). Z-ram shrinks embedded memory. Technical report, Microprocessor Report. `http://www.onversity.net/load/zram.pdf`.

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160.

Hammouri, G., Akdemir, K., and Sunar, B. (2009). Novel puf-based error detection methods in finite state machines. In *Information Security and Cryptology(ICISC) 2008*, pages 235–252.

Hsiao, M. Y. (1970). A class of optimal minimum odd-weight-column secded codes. *IBM Journal of R & D*, 14:395–401.

Irom, F. and Nguyen, D. (2007). Single event effect characterization of high density commercial NAND and NOR nonvolatile flash memories. *IEEE Transactions on Nuclear Science*, 54(6):2547–2553.

Johnston, A. H. (2000). Scaling and technology issues for soft error rates. 4th Annual Research Conference on Reliability. `http://trs-new.jpl.nasa.gov/dspace/handle/2014/16240`.

Joye, M. and Yen, S.-M. (2003). The montgomery powering ladder. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '02, pages 291–302, London, UK, UK. Springer-Verlag.

Jungnickel, D. and Pott, A. (1999). *Difference Sets, Sequences and Their Correlation Properties*, volume 542 of *NATO Science Series*, chapter Difference Sets: An Introduction. Kluwer Academic Publishers.

Karpovsky, M., Kulikowski, K. J., and Taubin, A. (2004). Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, pages 93–, Washington, DC, USA. IEEE Computer Society.

Karpovsky, M. and Nagvajara, P. (1989). Optimal codes for the minimax criteria on error detection. *IEEE Transaction on Information Theory*, 35(6):1299–1305.

Karpovsky, M. and Taubin, A. (2004). New class of nonlinear systematic error detecting codes. *IEEE Transactions on Information Theory*, 50(8):1818 – 1819.

Karpovsky, M. and Wang, Z. (2011). Algebraic manipulation detection codes resistant to advanced fault injection attacks. *IEEE Transactions on Information Theorem, submitted.*

Karpovsky, M. G., Kulikowski, K. J., and Wang, Z. (2007). Robust error detection in communication and computation channels. In *Workshop on Spectral Techniques.* `http://mark.bu.edu`.

Karpovsky, M. G., Kulikowski, K. J., and Wang, Z. (2008a). On-line self error detection with equal protection against all errors. *Interenational Journal of Highly Reliable Electronic System Design.* `http://mark.bu.edu`.

Karpovsky, M. G., Stankovic, R. S., and Astola, J. T. (2008b). *Spectral Logic and Its Applications in Design of Digital Devices.* John Wiley & Sons.

Karri, R., Wu, K., Mishra, P., and Kim, Y. (2002). Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 21(12):1509–1517.

Kermani, M. M. and Reyhani-Masoleh, A. (2008). A lightweight concurrent fault detection scheme for the AES S-boxes using normal basis. In *Cryptographic Hardware and Embedded Systems Workshop (CHES)*, pages 113–129.

Khursheed, S., Al-Hashimi, B., Reddy, S., and Harrod, P. (2009). Diagnosis of multiple-voltage design with bridge defect. *IEEE Transactions on Computers-Aided Design of Integrated Circuits and Systems*, 28(3):406 –416.

Kim, C. and Quisquater, J.-J. (2007). Fault attacks for CRT based RSA: New attacks, new results, and new countermeasures. In Sauveron, D., Markantonakis, K., Bilas, A., and Quisquater, J.-J., editors, *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, volume 4462 of *Lecture Notes in Computer Science*, pages 215–228. Springer Berlin / Heidelberg.

Kocher, P., Jaffe, J., and Jun, B. (1999). Differential power analysis. In *Lecture Notes in Computer Science*, pages 388–397.

Kocher, P. C. (1996). Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *KOBLITZ, N.(ed.) CRYPTO, Lecture Notes in Computer Science*, volume 1109, pages 104–113.

Krasniewski, A. (2008). Concurrent error detection for finite state machines implemented with embedded memory blocks of SRAM-based FPGAs. *Microprocessors and Microsystems*, pages 303–312.

Kulikowski, K., Venkataraman, V., Wang, Z., Taubin, A., and Karpovsky, M. (2008a). Asynchronous balanced gates tolerant to interconnect variability. In *IEEE International Symposium on Circuits and Systems, 2008. ISCAS 2008.*, pages 3190 –3193.

Kulikowski, K., Wang, Z., and Karpovsky, M. G. (2008b). Comparative analysis of fault attack resistant architectures for private and public key cryptosystems. In *Proceedings of International Workshop on Fault-tolerant Cryptographic Devices*, pages 41–50.

Kulikowski, K. J., Karpovsky, M. G., Taubin, E., Wang, Z., and Kulikowski, A. (2008c). Concurrent fault detection for secure QDI asynchronous circuits. In *Workshop on Dependable and Secure Nanocomputing (WDSN)*. `http:\\mark.bu.edu`.

Kulikowski, K. J., Venkataraman, V., Wang, Z., and Taubin, A. (2008d). Power balanced gates insensitive to routing capacitance mismatch. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '08, pages 1280–1285, New York, NY, USA. ACM.

Kulikowski, K. J., Venkataraman, V., Wang, Z., Taubin, A., and Karpovsky, M. G. (2008e). Asynchronous balanced gates tolerant to interconnect variability. In *IEEE International Symposium on Circuits and Systems (ISCAS 2008)*, pages 3190–3193. IEEE Press.

Lala, P. K. (1978). An adaptive double error correction scheme for semiconductor memory systems. *Digital processes*, 4:237–243.

Lala, P. K. (2001). *Self-Checking and Fault-Tolerant Digital Design, Book.* Elsevier.

Lala, P. K. (2003). A single error correcting and double error detecting coding scheme for computer memory systems. In *Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, DFT '03, pages 235–, Washington, DC, USA. IEEE Computer Society.

Lange, F. (1967). Correlation techniques. Princeton, N.J., Van Nostrand.

Lisbôa, C. A., Erigson, M. I., and Carro, L. (2007). System level approaches for mitigation of long duration transient in future technologies. In *12th IEEE European Test Symposium (ETS'07)*, pages 165–172.

Liu, W., Rho, J., and Sung, W. (2006). Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories. In *IEEE Workshop on Signal Processing Systems Design and Implementation, 2006. SIPS '06*, pages 303–308.

MacWilliams, F. and Sloane, N. (1998). *The Theory of Error-Correcting Codes.* North-Holland Mathematical Library.

Maistri, P. and Leveugle, R. (1982). Double-data-rate computation as a countermeasure against fault analysis. *IEEE Transactions on Computers*, 57(11):1528–1539.

Maistri, P., Vanhauwaert, P., and Leveugle, R. (2007). Evaluation of register-level protection techniques for the advanced encryption standard by multi-level fault injections. In Bolchini, C., Kim, Y.-B., Salsano, A., and Touba, N. A., editors, *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, pages 499–507. IEEE Computer Society.

Maiz, J., Hareland, S., Zhang, K., and Armstrong, P. (2003). Characterization of multi-bit soft error events in advanced srams. In *IEEE Int'l Electronic Device Meeting*, pages 519–522.

Malkin, T., Standaert, F.-X., and Yung, M. (2006). A comparative cost/security analysis of fault attack countermeasures. In Breveglieri, L., Koren, I., Naccache,

D., and Seifert, J.-P., editors, *Fault Diagnosis and Tolerance in Cryptography*, volume 4236 of *Lecture Notes in Computer Science*, pages 159–172. Springer Berlin / Heidelberg.

Massey, J. and Omura, J. (1986). Computational method and apparatus for finite field arithmetic. In *US Patent No. 4587627.*

Maxwell, M. S. (2005). *Almost Perfect Nonlinear functions and related combinatorial structures.* PhD thesis, Iowa State University.

Micheloni, R., Ravasio, R., Marelli, A., Alice, E., Altieri, V., Bovino, A., Crippa, L., Di Martino, E., D'Onofrio, L., Gambardella, A., Grillea, E., Guerra, G., Kim, D., Missiroli, C., Motta, I., Prisco, A., Ragone, G., Romano, M., Sangalli, M., Sauro, P., Scotti, M., and Won, S. (2006). A 4Gb 2b/cell NAND flash memory with embedded 5b BCH ECC for 36mb/s system read throughput. In *IEEE International Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers*, pages 497–506.

Micron (2008). *Wear-leveling techniques in NAND flash devices.* Micron. `http://download.micron.com/pdf/technotes/nand/tn2942_nand_wear_leveling.pdf`.

Mollard, M. (1986). A generalized parity function and its use in the construction of perfect codes. In *SIAM J. Algebraic Discrete Methods*, volume 7, pages 113–115.

Monnet, Y., Renaudin, M., Leveugle, R., Clavier, C., and Moitrel, P. (2006). Case study of a fault attack on asynchronous DES crypto-processors. In Breveglieri, L., Koren, I., Naccache, D., and Seifert, J.-P., editors, *Fault Diagnosis and Tolerance in Cryptography*, volume 4236 of *Lecture Notes in Computer Science*, pages 88–97. Springer Berlin / Heidelberg.

Moore, S. K. (2007). Masters of memory. *IEEE Spectrum*, 44(1):45–49.

Moore, S. W., Anderson, R. J., Cunningham, P., Mullins, R., and Taylor, G. (2002). Improving smart card security using self-timed circuits. In *8th International Symposium on Asynchronous Circuits and Systems (ASYNC 2002)*, pages 211–218. IEEE Computer Society.

Nangate (2011). Nangate 45nm open cell library. `http://www.nangate.com`.

Pei, T.-B. and Zukowski, C. (1992). High-speed parallel CRC circuits in VLSI. *IEEE Transactions on Communications*, 40(4):653–657.

Penzo, L., Sciuto, D., and Silvano, C. (1995). Construction techniques for systematic SEC-DED codes with single byte error detection and partial correction capability for computer memory systems. *IEEE Transactions on Information Theory*, 41(2):584 –591.

Phelps, K. T. (1983). A combinatorial construction of perfect codes. In *SIAM J. Algebraic Discrete Methods*, volume 4, pages 398–403.

Phelps, K. T. and Levan, M. (1995). Kernels of nonlinear Hamming codes. *Designs, Codes and Cryptography*, 6:247–257.

Piret, G. and Quisquater, J.-J. (2003). A differential fault attack technique against spn structures, with application to the AES and KHAZAD. In *Cryptographic Hardware and Embedded Systems Workshop (CHES)*, pages 77–88.

Rao, T. and Garcia, O. (1971). Cyclic and multiresidue codes for arithmetic operations. *IEEE Transactions on Information Theory*, 17(1):85 – 91.

Rao, T. R. N. and Fujiwara, E. (1989). *Error-control coding for computer systems.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Reddy, S. (1978). A class of linear codes for error control in byte-per-card organized digital systems. *IEEE Transactions on Computers*, C-27:455–459.

Reyhani-Masoleh, A. and Hasan, M. A. (2002). A new construction of Massey-Omura parallel multiplier over GF(2^m). *IEEE Transactions on Computers*, 51(5):511–520.

Rizwan, S. (2008). Retimed decomposed serial Berlekamp-Massey (BM) architecture for high-speed Reed-Solomon decoding. In *21st International Conference on VLSI Design, 2008. VLSID 2008*, pages 53–58.

Roberts, D., Austin, T., Blauww, D., Mudge, T., and Flautner, K. (2005). Error analysis for the support of robust voltage scaling. In *Sixth International Symposium on Quality of Electronic Design, 2005. ISQED 2005.*, pages 65 – 70.

Rochet, R., Leveugle, R., and Saucier, G. (1993). Analysis and comparison of fault tolerant FSM architecture based on SEC codes. In *The IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*, pages 9–16.

Samyde, D., Skorobogatov, S., Anderson, R., and jacques Quisquater, J. (2002). On a new way to read data from memory. In *Proceedings of the First International IEEE Security in Storage Workshop*, pages 65–, Washington, DC, USA. IEEE Computer Society.

Santhi, N. (2007). On algebraic decoding of q-ary Reed-Muller and product Reed-Solomon codes. In *IEEE International Symposium on Information Theory, 2007. ISIT 2007*, pages 1351 –1355.

Sarwate, D. V. and Shanbhag, N. R. (2001). High-speed architectures for Reed-Solomon decoders. *IEEE Transactions on Very Large Scale Integrated Systems*, 9(5):641–655.

Satoh, S., Tosaka, Y., and Wender, S. A. (2000). Geometric effect of multiple-bit soft errors induced by cosmic ray neutrons on DRAMs. *IEEE Electron Device Letters*, 21(6):310 –312.

Schmidt, J.-M. and Herbst, C. (2008). A practical fault attack on square and multiply. In *Proceedings of the 2008 5th Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 53–58, Washington, DC, USA. IEEE Computer Society.

Schmidt, J. M. and Hutter, M. (2007). Optical and EM fault-attacks on CRT-based RSA: Concrete results. In *15th Austrian Workshop on Microelectronics*. `http://events.iaik.tugraz.at/austrochip2007/proceedings/index.htm`.

SECG (2000). Standards for efficient cryptography, sec 2: Recommended elliptic curve domain parameters. `http://www.secg.org/collateral/sec2_final.pdf`.

Seth, K., Viswajith, K., Srinivasan, S., and Kamakoti, V. (2006). Ultra folded high-speed architectures for Reed Solomon decoders. In *VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design., 19th International Conference on*, pages 4 pp.–.

Sherman, H. (1956). Some optimal signals for time measurement. *IRE Transactions on Information Theory*, IT-2(1).

Skorobogatov, S. (2010). Optical fault masking attacks. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2010 Workshop on*, pages 23 –29.

Skorobogatov, S. P. and Anderson, R. J. (2003). Optical fault induction attacks. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 2–12, London, UK. Springer-Verlag.

Sudan, M. (1997). Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity.*, 13(1):180–193.

Sun, F., Devarajan, S., Rose, K., and Zhang, T. (2007). Design of on-chip error correction systems for multilevel NOR and NAND flash memories. *IET Circuits, Devices and Systems*, 1(3):241–249.

Sunar, B., Gaubatz, G., and Savas, E. (2007a). Sequential circuit design for embedded cryptographic applications resilient to adversarial faults. *IEEE Transactions on Computers*, 57:126–138.

Sunar, B., Martin, W. J., and Stinson, D. R. (2007b). A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Transactions on Computers*, 56(1):109–119.

Swift, G. M. (2001). In-flight observations of multiple-bit upset in DRAMs. *IEEE Transactions on Nuclear Science*, 47(6):2386–.

Tahir, J. M., Dlay, S. S., Naguib, R. N. G., and Hinton, O. R. (1995). Fault tolerant arithmetic unit using duplication and residue codes. *The VLSI Journal on Integration*, 18(2-3):187 – 200.

Tam, S. (2006). *Application Note: Single Error Correction and Double Error Detection.* XILINX. `http://www.xilinx.com/support/documentation/application_notes/xapp645.pdf`.

Trichina, E. and Korkikyan, R. (2010). Multi fault laser attacks on protected CRT-RSA. *Workshop on Fault Diagnosis and Tolerance in Cryptography*, 0:75–86.

Vasil'ev, J. L. (1962). On nongroup close-packed codes. In *Problemy Kibernetiki. (in Russian)*, volume 8, pages 337–339.

Vasyltsov, I., Hambardzumyan, E., Kim, Y.-S., and Karpinskyy, B. (2008). Fast digital TRNG based on metastable ring oscillator. In *In Proceedings of Cryptographic Hardware and Embedded Systems Workshop (CHES)*, pages 164–180.

Wallace, C. S. (1964). A suggestion for a fast multiplier. *IEEE Transactions on Electronic Computers*, EC-13(1):14 –17.

Wang, Z. and Karpovsky, M. (2010). Robust FSMs for cryptographic devices resilient to strong fault injection attacks. In *16th IEEE International On-Line Testing Symposium (IOLTS'2010)*, pages 240–245.

Wang, Z. and Karpovsky, M. (2011). Algebraic manipulation detection code and their applications for design of secure cryptographic devices. In *IEEE International On-Line Testing Symposium (IOLTS), submitted.*

Wang, Z., Karpovsky, M., and Joshi, A. (2011a). Influence of metallic tubes on the reliability of CNTFET SRAMs: Error mechanisms and countermeasures. In *Great Lakes Symposium on VLSI (GLSVLSI), to appear.*

Wang, Z., Karpovsky, M., and Joshi, A. (2011b). Reliable NAND flash memories based on nonlinear multi-error correcting codes. *IEEE Transactions on Very Large Scale Integration Systems, revised,.*

Wang, Z., Karpovsky, M., and Joshi, A. (2011c). Secure multipliers resilient to strong fault injection attacks based on multilinear arithmetic codes. *IEEE Transactions on Very Large Scale Integration Systems, accepted.*

Wang, Z., Karpovsky, M., and Kulikowski, K. (2010a). Design of memories with concurrent error detection and correction by nonlinear SEC-DED codes. *Journal of Electronic Testing*, 26:559–580. 10.1007/s10836-010-5168-5.

Wang, Z., Karpovsky, M., and Sunar, B. (2009a). Multilinear codes for robust error detection. *IEEE International On-Line Testing Symposium*, pages 164–169.

Wang, Z., Karpovsky, M., Sunar, B., and Joshi, A. (2009b). Design of reliable and secure multipliers by multilinear arithmetic codes. In *Information and Communications Security*, volume 5927 of *Lecture Notes in Computer Science*, pages 47–62.

Wang, Z., Karpovsky, M. G., and Joshi, N. (2010b). Reliable MLC NAND flash memories based on nonlinear t-error-correcting codes. In *2010 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010)*, pages 41–50. IEEE Computer Society.

Wang, Z., Karpovsky, M. G., and Kulikowski, K. (2009c). Replacing linear Hamming codes by robust nonlinear codes results in a reliability improvement of memories. In *IEEE/IFIP International Conference on Dependable Systems and Networks, 2009. DSN '09.*, pages 514–523.

Whitaker, S., K.Cameron, G.Maki, J.Canaris, and P.Owsley (1991). VLSI Reed-Solomon processor for the hubble space telescope. In *VLSI Signal Processing IV IEEE Press.*

Yaakobi, E., Ma, J., Caulfield, A., Grupp, L., Swanson, S., Siegel, P. H., and Wolf, J. K. (2009). Error correcting coding for flash memories. Technical report, Center for Magnetic Recording Research (CMRR).

Yuan, J., Carlet, C., and Ding, C. (2006). The weight distribution of a class of linear codes from perfect nonlinear functions. *IEEE Transactions on Information Theory*, 52(2):712–717.

CURRICULUM VITAE

Zhen Wang

Zhen Wang was born in Qingdao – a beautiful coast city in China. He received his B.S and M.S degree in Information and Communication Engineering from Zhejiang University, China in 2006. Since the summer of 2006, he started working towards his PhD degree at Boston University under the supervision of Prof. Mark G. Karpovsky and Prof. Ajay Joshi. His research is focused on the design of robust codes and the application of robust codes in building reliable and secure devices, e.g. secure cryptographic devices and reliable memories. He also conducts research to investigate the influence of nano-scale technologies on the reliability of modern digital systems, etc. He is now working in the DSP simulator group in Mediatek, Inc.