# Hardware implementation of secure Shamir's secret sharing scheme

Pei Luo[1], Andy Yu-Lun Lin[1], Zhen Wang[2], and Mark Karpovsky[1]

[1] Reliable Computing Lab, Electrical and Computer Engineering, Boston University
luopei@bu.edu, ayllin@bu.edu, markkar@bu.edu
[2] Mediatek Wireless, Inc
wang.zhen.mtk@gmail.com

**Abstract.** Shamir's secret sharing scheme is an effective way to distribute secret to a group of shareholders. But this scheme is vulnerable to cheaters and attackers and thus how to protect the system from cheating and attacks is a big problem. In this paper, we proposed to use robust codes and algebraic manipulation detection (AMD) codes to protect the secret sharing module. Simulation and synthesis results show that the proposed architecture can improve the security level significantly even under strong cheating and attack models with some extra area and timing overheads.

## 1 Introduction

A secret sharing scheme is a method of hiding a secret among several shadows such that the secret can be retrieved only by some subsets of these shadows. Secret sharing is useful in many cryptographic applications, especially in sharing the key among some shareholders [1,2,3,4].

In 1979, Blakley [1] and Shamir [2] independently introduced secret sharing schemes. In a secret sharing scheme, a secret $S$ is divided into $L$ shares and distributed to $L$ shareholders by a trusted dealer. The shared secret $S$ can only be recovered when $l$ ($l \leq L$) or more than $l$ shares are available. Such a scheme is called an $(l, L)$ secret sharing.

Shamir's secret sharing scheme [2] is not immune to cheating and other kinds of attacks. If there are dishonest shareholders, they could purposely distort their shares such that honest participants can only obtain a fake secret. What's more, the attacker can retrieve the real secret based on the fake secret and the errors injected [3,4]. Therefore, in order to protect the cryptographic devices against the attackers, it is important to design a scheme to protect the secret sharing module such that it is still secure even under cheating and other kinds of attacks.

In prior works, various methods have been proposed to protect the secret sharing schemes against the attacks and cheating. In [3], Tompa and Woll showed that the cheater may cheat successfully with a high probability in Shamir's secret sharing scheme. They made some modification to Shamir's scheme and the successful cheating probability can be reduced dramatically. The authors in [4] proposed to use a scheme that takes more than $l$ participants to decode the secret for a $(l, L)$ secret sharing scheme thus to identify the cheaters. In [5], the authors proposed to use the idea of sub-secrets and sub-shares distributed by the shareholders such that each shareholder is also a dealer.

At the same time, some other secret sharing schemes resistant to cheating are also proposed. In [6], the authors proposed to apply a one-way hashing function along with the use of arithmetic coding to detect cheating and identify the cheaters. In [7], the authors proposed a computationally secure scheme based on RSA assumption for secret sharing.

Previous papers also deal with cheater identification in different secret sharing schemes [4,7,8,9]. There are several methods for cheater identification. The dealer can generate and distribute additional information, such as using check vectors and certificate vectors for each shareholder. Error-correcting codes can also be introduced for cheater detection and identification such that faked shares can be treated as error codes to be detected and corrected [4,10,11]. In [4], the authors proposed to use more than $l$ shareholders during the secret reconstruction thus the redundant shares can be used for cheater detection and identification in a $(l, L)$ Shamir's secret sharing schemes.

In this paper, we propose to use error detection codes (EDC) to detect cheating and identify the cheaters thus to improve the security level of the system. At the same time, the proposed architecture can also detect most of the fault injection attacks with very low probability of missing an error. Cheater identification methods for the proposed schemes are also discussed in this paper. The proposed schemes will not change the basic conception and operation of Shamir's secret sharing scheme and thus it's compatible with the previous schemes based on Shamir's secret sharing.

We construct the secret reconstruction and error detection modules in Verilog and synthesize in Cadence Encounter RTL Compiler with the Nangate 45nm Opencell library version v2009_07. We run simulation for different cheating models and inject faults into the system to test the security level. The results show that the proposed schemes can detect the cheating and attacks with a very high probability while incur slightly larger timing and area overhead than the non-protected scheme at the same time.

The rest of this paper is organized as following: In Section 2, basic conceptions of Shamir's secret sharing will be reviewed, and several different kinds of cheating models used in this paper will be introduced. In Section 3, we will discuss the architecture of the proposed schemes based on roust and AMD codes and discuss their security level under different cheating models. In Section 4, we will discuss the cheater identification for the proposed schemes and compare our methods with previous ones. In Section 5, cheating and fault injection simulations are implemented to verify the security level of the proposed schemes. Synthesis results are given to evaluate the area and timing overheads of the proposed architectures.

## 2 Basic conceptions

### 2.1 Shamir's $(l, L)$ secret sharing

In a Shamir's $(l, L)$ secret sharing [2], there are $L$ shareholders $P = \{P_0, P_1, \cdots, P_{L-1}\}$ and a trusted dealer $D$. We use the shareholders' IDs $(z_0, z_1, \cdots, z_{L-1})$ to denote each participant. Secret is generated and distributed to shareholders by the dealer $D$. Secret can be reconstructed based on Lagrange interpolation polynomial by taking any $l$ shares $(\alpha_{i_0}, \cdots, \alpha_{i_{l-1}})$ of participants and their IDs $(z_{i_0}, \cdots, z_{i_{l-1}})$ where $\{i_0, \cdots, i_{l-1}\} \subseteq \{0, 1, \cdots, L-1\}$ [2,3,4].

Shamir's secret sharing scheme provides a method of hiding this secret such that any $l$ or more participants would be able to reconstruct the original secret but any less than $l$ participants would not be able to reconstruct the original secret. Assume all the computation are in $GF(2^n)$ $(L \leq 2^n)$, in which $n$ is the number of bits of the secret, Shamir's secret sharing scheme consists of two algorithms [2,4]:

1. *Share generation algorithm* Construct a polynomial $\alpha(z) = S_0 + S_1 z + S_2 z^2 + \cdots + S_{l-1} z^{l-1}$ where $S_i (i \in \{0, 1, \cdots, l-1\})$ and $z$ belong to $GF(2^n)$. $S_i (i \in \{0, 1, \cdots, l-2\})$ are randomly generated in $GF(2^n)$ and $S_{l-1}$ is the secret. Each shareholder with ID $z_i$ receives a share $\alpha(z_i)$ and we assume the IDs $z_i$ are publicly known and unique for each different shareholder. We denote $\alpha(z_i)$ as $\alpha_i$, and $z_{i_j}$ as $z_j$ for simplicity in this paper.
2. *Secret reconstruction algorithm* For a polynomial of degree $l-1$, with knowledge of at least $l$ data points, we can reconstruct the exact polynomial using Lagrange interpolation in $GF(2^n)$ and thus reconstruct the secret $S_{l-1}$.

In this paper, we choose $S_{l-1}$ as the secret for simplicity, and thus the equation for secret reconstruction is as following:

$$S_{l-1} = \sum_{i=0}^{l-1} \frac{\alpha_i}{\prod_{j=0, j \neq i}^{l-1} (z_i + z_j)}, \tag{1}$$

and equivalently

$$S_{l-1} = c_0 \alpha_0 + c_1 \alpha_1 + \cdots + c_{l-1} \alpha_{l-1}, \tag{2}$$

where $c_i = \prod_{j=0, j \neq i}^{l-1} (z_i + z_j)^{-1}$. For a shareholder $z_i$, we can represent the cheating performed by the shareholders as $\tilde{\alpha}_i = \alpha_i + e_i$, where $e_i$ is the error injected by $z_i$. A cheater can inject errors into his share and generate a fake secret $\tilde{S}_{L-1} = S_{L-1} + e$ and he can precisely control the error $e$ when the IDs $z_i$ are publicly known.

### 2.2 Cheating models

For Shamir's secret sharing, we assume that the dealer is honest and the shares distributed to the shareholders are not distorted. According to the definition of Shamir's secret sharing and the description above, any of these $l$ shareholders can distort the secret by submitting a false share. Accodring to the discussion in Section 2.1, a cheater can inject errors into his share and generate a fake secret $\widetilde{S}_{L-1} = S_{L-1} + e$ and he can precisely control the error $e$ when the IDs $z_i$ are publicly known. Then the attacker can calculate the real secret using the fake secret according to (1).

Thus we must protect the secret sharing system against the cheaters and in this section we will deal with several different kinds of strong cheating models to improve the security level of the proposed schemes. We assume that up to $l$ shareholders can cooperate to generate a fake secret.

At the same time, we also deal with two different cases about whether the cheaters can get information of the reconstructed secret. In the first case, the output of the secret reconstruction module is hidden from the participants and thus the participants have no knowledge of the reconstructed secret; in the other case, the shareholders are able to gain knowledge of the output, thus they can try to cheat according to the information they get.

In order to emulate the real cheating situations and estimate the security level of the proposed schemes, we will consider several different kinds of cheating models in this design:

- *Type 1* Less than $l$ shareholders are cheating and the output of the secret reconstruction module is hidden from the participants;
- *Type 2* Less than $l$ shareholders are cheating but there is a feedback from the output such that the cheaters can get information of the secret to help them to generate a fake secret in the next round;
- *Type 3* All of the $l$ shareholders are cheating and the output of the secret reconstruction module is hidden from the participants;
- *Type 4* All of the $l$ shareholders are cheating and there is a feedback from the output such that the cheaters can gain some knowledge of the secret to help them to cheat in the next round.

Besides the cheating models above, we assume that the attackers can also inject faults into the hardware of the system to affect the working of the circuits. Advanced attackers can cause stuck-at-1 and stuck-at-0 faults or flip the output of the gates [12,13,14]. Our architecture can detect such fault injection attacks with very high probability.

## 3 Construction of the proposed schemes

In this section we will describe the proposed architecture which is resistant to stronger cheating models and fault injection attacks. The proposed architecture is composed of two parts, the first stage is the secret reconstruction module implementing an $(l, L)$ Shamir's secret sharing scheme and the second stage is the error checking module shown as in Figure 1.

Different from the previous schemes, we divide the output of the secret reconstruction module into two parts, the real secret $y$ and the redundant part $R$, and thus the original output of the secret reconstruction module is $S_{l-1} = \{y, R\}$ for our proposed schemes. In this paper, we denote the size of $y$ as $k$ ($y \in GF(2^k)$), and the size of $R$ as $r$ ($R \in GF(2^r)$), and the size of $S_{l-1}$ as $n = k + r$ ($S_{l-1} \in GF(2^n)$). For an error detection code $C$, $S_{l-1}$ is a codeword of $C$ in the proposed schemes ($S_{l-1} \in C$), and the error checking module will verify if the reconstructed secret $S_{l-1} \in C$ or not.

For the error checking stage, the reconstructed secret may be distorted and we denote it as $\tilde{S}_{l-1} = \{\tilde{y}, \tilde{R}\}$, and $\tilde{S}_{l-1}$ will be input into the error checking module. The error checking module then checks if $\tilde{S}_{l-1}$ is a codeword of $C$. The *error* signal is set and there will be no output at the secret output port when errors are detected in $\tilde{S}_{l-1}$.

In this section, different kinds of error detection codes will be implemented for detection of cheating and the results will be compared in Section 5 to find the best solution for the protection of secret sharing under different cheating models.

In this paper, we assume the size of the unprotected secret is 96 bits for the original secret sharing module without redundancy, which means $k = 96$ and $y \in GF(2^{96})$. For the architecture with error

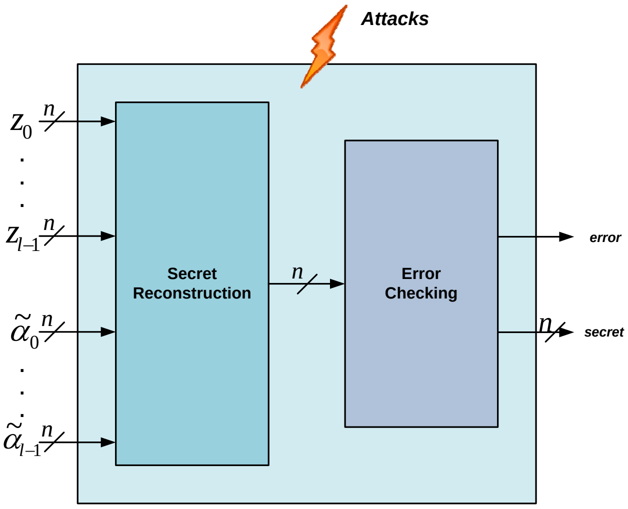

**Fig. 1.** Architecture of the secret sharing scheme

detection module, we assume the size of redundancy is 32 bits, $r = 32$ and $R \in GF(2^{32})$. Thus the size of the output of the proposed scheme is $n = k + r = 128$. The redundancy will increase the security of the target platform significantly while incur some extra area and timing overhead.

### 3.1 Why Linear code is not enough

In this section, we will discuss the secure secret sharing scheme based on linear codes. For the system based on linear code $C$, the secret $S_{l-1}$ generated by the dealer should be a codeword of $C$ ($S_{l-1} \in C$). In secret reconstruction stage, errors can be injected into the shares by the shareholders, thus to generate a fake secret $\tilde{S}_{l-1}$, $\tilde{S}_{l-1} = S_{l-1} + e$. For linear codes, the error is masked or undetected if $\tilde{S}_{l-1}$ is also a codeword of $C$ ($\tilde{S}_{l-1} \in C$).

For a linear $(n, k)$ code $C$ used in this architecture, $k$ is the number of information bits (secret), and $n$ is the size of the codeword including both the information bits and the redundancy.

If the attacker has no knowledge of the code used in this architecture, he will successfully cheat the system with a probability of $\frac{2^k}{2^n} = 2^{-r}$ in the worst case assuming that he injects error $e$ in $S_{l-1}$ with the same probability. However, one major weakness of linear codes used for security is that the sum of any two codewords is also a codeword. Therefore if the attacker has knowledge of the code $C$ used in this architecture, he can bypass the detection with a probability 1 by injecting an error $e_i$ into $s_i$ such that $e \in C$. Thus the secret sharing scheme based on linear code is not secure enough against strong attackers.

For example, if $z_0$ is cheating, then he can inject an error $e_0$ into his share such that $e = c_0 e_0 \in C$ (see (2)). So, with the knowledge of the code $C$ used, any attacker out of $l$ participants can cheat the system into reconstructing a wrong secret and bypass the detection with a probability of 1.

*Example 1.* Use a (5,2,3) Hamming code $C = \{00000, 11001, 10111, 01110\}$ and $l = 3$ as a simple example. For the secret generation stage, denote the secret distribution polynomial as $\alpha(z) = S_0 + S_1 z + S_2 z^2$, where $S_2$ is the secret. If we assume the attacker knows the code that is being used, he also knows the equation for reconstructing the secret is

$$\alpha_0 \frac{1}{(z_0 + z_1)(z_0 + z_2)} + \alpha_1 \frac{1}{(z_0 + z_1)(z_1 + z_2)} + \alpha_2 \frac{1}{(z_0 + z_2)(z_1 + z_2)} = S_2. \tag{3}$$

Assume the attacker $z_0$ can inject error $e_0$ into his share, the resulting output becomes

$$\begin{aligned}
&\tilde{\alpha}_0 \frac{1}{(z_0+z_1)(z_0+z_2)} + \alpha_1 \frac{1}{(z_0+z_1)(z_1+z_2)} \\
&+ \alpha_2 \frac{1}{(z_0+z_2)(z_1+z_2)} = (\alpha_0 + e_0) \frac{1}{(z_0+z_1)(z_0+z_2)} \\
&+ \alpha_1 \frac{1}{(z_0+z_1)(z_1+z_2)} + \alpha_2 \frac{1}{(z_0+z_2)(z_1+z_2)} \\
&= S_2 + e_0 \frac{1}{(z_0+z_1)(z_0+z_2)},
\end{aligned} \tag{4}$$

where $e = e_0 \frac{1}{(z_0+z_1)(z_0+z_2)}$.

The probability of missing an error is $\frac{1}{2^3}$ if the attacker has no information about the system. However, since $z_0, z_1$ and $z_2$ are publicly known, in order to bypass the protection by linear codes, the attacker only has to find out $e_0$ such that $e \in C$. By injecting this $e_0$, the attacker can guarantee that his cheating will be successful with probability 1.

### 3.2 Detection with Robust code

According to the discussion in Section 3.1, it's obvious that linear codes are insufficient for the protection of a secure secret sharing module under strong cheating models. In order to improve the security level of the system, we will use robust codes [15,16,17] to protect the secret sharing module in this section.

For robust codes, the codewords have the form of $(y, f(y))$ $y \in GF(2^k), f(y) \in GF(2^r)$. Error $e = (e_y, e_f)$, $(e_y \in GF(2^k)), e_f \in GF(2^r)$ is masked for a given $y$ if

$$f(y + e_y) = f(y) + e_f. \tag{5}$$

For good robust codes, the fraction of $y$ satisfying (5) is very small for any error $(e_y, e_f)$ [15,16,17].

For the system with $k = 96$ information bits based on robust codes, we divide $y$ into 3 parts $(y_0, y_1, y_2)$ and $y_i \in GF(2^{32}), i = 1, 2, 3$. We use modified robust quadratic code

$$R = f(y) = y_0 y_1 + y_2^3 \tag{6}$$

in this paper. This modified robust quadratic code contains 96 information bits $(y_0, y_1, y_2)$ $(y_0, y_1, y_2 \in GF(2^{32}))$ and 32 redundant bits $f(y)$ $(f(y) \in GF(2^{32}))$.

If error $(e_0, e_1, e_2, e_f)$ $(e_i, e_f \in GF(2^{32}))$ is injected, the error masking equation is

$$(y_0 + e_0)(y_1 + e_1) + (y_2 + e_2)^3 = f(y) + e_f, \tag{7}$$

then

$$y_0 e_1 + y_1 e_0 + y_2^2 e_2 + y_2 e_2^2 + e_2^3 + e_0 e_1 + e_f = 0. \tag{8}$$

Thus the probability of a nonzero error being masked assuming equiprobable $y$ is $Q_{rq}(e) = Q_{rq}(e_0, e_1, e_2, e_f) = 2^{-r+1} = 2^{-31}$ [15,16,17].

*Example 2.* In this example, we will show why Robust code is sufficient for *Type 1* cheating model, but fails for *Type 2-4* models. Under *Type 1* cheating model, the attacker does not know $S_2 = (y, f(y))$ and thus cannot predict an error that can be injected without being detected with certainty.

In this example, we will use a system with 9 bits original secret and 3 bits redundancy to explain the robust code based system. The polynomial for $GF(2^3)$ is $f(x) = x^3 + x + 1$ and $f(x) = x^{12} + x^3 + 1$ is the primitive polynomial for $GF(2^{12})$. If we take $y = (010111101)$, in which $y_0 = (010), y_1 = (111)$ and $y_2 = (101)$, then $f(y) = y_0 y_1 + y_2^3 = (011)$. Thus the secret with redundancy is $S_2 = (010111101011)$. For this robust code based system, any random error will have a probability of being undetected of $2^{-r+1} = \frac{1}{4}$.

In *Type 2* cheating model, if the attacker can obtain information of $S_2$ from the feedback, they can calculate the error to inject thus to produce an undetectable error. For example, if $z_0 = (100011111110), z_1 = (001010000110), z_2 = (011001110101)$ are the participants during the secret reconstruction, and $z_0$ is the cheater with knowledge of $S_2$, he can inject an error $e_0$ such that $\frac{1}{(z_0+z_1)(z_0+z_2)}e_0 + S_2 \in C$ to bypass the protection of the error checking module based on robust code, where $C$ is the set of codewords of the robust code in equation (6). Since $z_0, z_1, z_2, S_2, C$ are all known, it is easy to work out the error $e_0$ which will cause an undetectable error.

For instance, to distort the secret into $\tilde{S}_2 = (011100011101)$, the attacker needs to introduce an error such that $e = (001011110110) = e_0 \frac{1}{(z_0+z_1)(z_0+z_2)}$. It is not difficult for $z_0$ to find out that an error $e_0 = (101001111110)$ can be introduced into his share to achieve this.

Similarly, under *Type 3* cheating model, the $l$ shareholders can collaborate to work out the secret $S_2$ and thus inject undetectable errors into the system. Once $S_2$ is known, it will be easy for the attackers to inject an error which can bypass the error checking module. Under *Type 4* cheating model, the attackers can get information of $S_2$ either from the feedback or collaboration. So the robust code based scheme is not secure under *Type 2-4* cheating models.

### 3.3 Detection with AMD code

In this section, we will construct an architecture based on algebraic manipulation detection (AMD) codes [18,19,20,21,22] and discuss the proposed architecture under *Type 2-4* cheating models.

For AMD code, a random vector $x \in GF(2^m), m = tr, t \in (1, 2, 3, \cdots)$ will be generated and stored securely in the device such that it cannot be read by the attacker without destroying the device itself.

It is generally not a good idea to store the information bits $y$ in the secure memory since if an attacker reads the secure memory by destroying it, they can obtain knowledge of the secret $y$. This would allow outside attackers to gain access to the secret without knowledge of the secret sharing scheme itself. However, if we only store the random bits $x$ in the secure memory, even if this memory is read and the attackers gain knowledge of $x$ by destroying the secret sharing module, our secret still remains secure.

The codewords $(y, x, f(x, y))$ of AMD consist of three parts, the message $y \in GF(2^k)$, random vector $x \in GF(2^m)$, and redundant bits $f(x, y) \in GF(2^r)$. Since $x$ is stored in the system and thus the secret $S_{l-1}$ is $(y, f(x, y))$. For this architecture, when the secret is reconstructed, the device will retrieve $x$ from the secure memory and check if

$$f(\tilde{x}, \tilde{y}) = \tilde{f}(x, y) \tag{9}$$

where $\tilde{y}$ and $\tilde{f}(x, y)$ are from the distorted secret, $\tilde{x}$ is the distorted random number $x$.

With knowledge of $y$ and $f(x, y)$, as well as the polynomial used to construct $f(x, y)$, the attackers could work out a set of possible values for $x$. If $x \in GF(2^m), m = r$, and $f(x, y) = \sum_{i=1}^{b} y_i x^i + x^{b+2}$ where $y_i, x \in GF(2^r)$ and $y = (y_1, \cdots, y_b)$, then the size of the set of possible $x$ is at most $b+2$ [18,19,20]. To increase the security level of our system, the size of the random number in this system can be $m = tr$ where $t \geq 2$ [18,19,20]. Here we use AMD code with $t = 2$ and thus $x = (x_0, x_1)$, $x_0, x_1 \in GF(2^{32})$. Then the size of the set of possible $x = (x_0, x_1)$ will be much larger and thus improve the security level of the system. The same as robust code based scheme, $y$ is also divided into three parts $(y_0, y_1, y_2)$ $(y_0, y_1, y_2 \in GF(2^{32}))$ in AMD code based system. Thus the encoding polynomial used for AMD code in this system is

$$f(x_0, x_1, y) = y_0 x_0 + y_1 x_1 + y_2 x_0 x_1 + x_0^3 + x_1^3, \tag{10}$$

where $y = (y_0, y_1, y_2)$, assume the attackers can inject errors into their shares and be able to inject faults into the hardware system as well to distort $x_0, x_1$. Assume the error for a set of $(y_0, y_1, y_2, x_0, x_1, f)$ is $(e_0, e_1, e_2, e_{x_0}, e_{x_1}, e_f)$, then the error masking equation is:

$$\begin{aligned}
&(y_0 + e_0)(x_1 + e_{x_1}) + (y_1 + e_1)(x_1 + e_{x_1}) \\
&+(y_2 + e_2)(x_0 + e_{x_0})(x_1 + e_{x_1}) + (x_0 + e_{x_0})^3 \\
&+(x_1 + e_{x_1})^3 = f(x_0, x_1, y) + e_f.
\end{aligned} \tag{11}$$

Note that all the computation are in $GF(2^r)$, Equivalently

$$\begin{aligned}&y_0e_{x_0}+x_0e_0+e_0e_{x_0}+y_1e_{x_1}+x_1e_1+e_1e_{x_1}\\&+y_2x_0e_{x_1}+y_2x_1e_{x_0}+y_2e_{x_0}e_{x_1}+e_2x_0x_1\\&+e_2x_0e_{x_1}+e_2e_{x_0}x_1+e_2e_{x_0}e_{x_1}+x_0^2e_{x_0}+e_{x_0}^2x_0\\&+e_{x_0}^3+x_1^2e_{x_1}+e_{x_1}^2x_1+e_{x_1}^3+e_f=0.\end{aligned} \tag{12}$$

Since $y$ is known and the errors are injected and controlled by the attackers, (12) can be rewritten as

$$x_0^2e_{x_0}+x_1^2e_{x_1}+e_2x_0x_1+Ax_0+Bx_1+C=0, \tag{13}$$

where $A, B, C$ are constants based on $y$ and the errors according to (12). Since the attackers have no knowledge of $x_0$ and $x_1$, we have from (13) that the probability of successful attacks is at most $2^{-r+1}$

*Example 3.* Let $S_2=(y_0,y_1,y_2,R)$ and random bits $x_0$, $x_1$ are stored securely in hardware. Choose $(y_0,y_1,y_2)=(001001100)$ and $x_0=(010)$, $x_1=(110)$, then $R=f(x_0,x_1,y)=(001)$. $S_2=(001001100001)$. The attackers wish to choose an error $e$ such that $\tilde{S}_2=S_2+e\in C$. To do this with complete certainty, the attackers need knowledge of $x_0, x_1$ to determine valid codewords in $C$. In the worst case scenario, the attackers need to generate a list of all sets of $(x_0,x_1)$ for the specific secret $S_2$. In this case, there are 5 sets of $(x_0,x_1)$ which satisfy the parameters above: $(001,101)$, $(010,110)$, $(100,101)$, $(011,111)$, $(101,101)$. Thus, the probability of successfully cheating is $\frac{1}{5}$, which is less than the upper bound $2^{-r+1}=\frac{1}{4}$ [18,19,20].

Since random bits $x_0$ and $x_1$ are stored securely, the attacker has no knowledge of these information. Without knowing $x_0$ and $x_1$, the attackers cannot guarantee a successful undetected attack in our system.

So according to the above discussion, the AMD based system has very high security level even under *Type 2-4* cheating models. The proposed AMD based architecture will still maintain a very high security level even under strong cheating models and threatened by fault injection attacks.

## 4 Cheater identification

As opposed to those approaches described in [8,7,4,9], we will present an approach for identification of cheaters in this section. This approach will provide for an efficient hardware implementations for the case when the number of cheaters $m$ is less than $l$.

This approach is based on an increase in the number of participants from $l$ to $N=l+I$ and computing not one (probably distorted) secret, but $A$ secrets based on $A$ different subsets of $l$ shareholders out of the set of $N$ shareholders. Let's construct an $(A\times N)$ matrix $M=(M_{ij})$ such that $M_{ij}=1$ if shares of participant $i$ $(i=0,1,\cdots,N-1)$ are used to reconstruct the secret $j$ $(j=0,1,\cdots,A-1)$. We note that the number of 1s in very row of $M$ is exactly $l$.

*Example 4.* Let $m=1, l=4$ and $N=5$, then we can select the matrix $M_1$ as

$$M_1=\begin{vmatrix}0&1&1&1&1\\1&0&1&1&1\\1&1&0&1&1\\1&1&1&0&1\end{vmatrix}.$$

Then the first secret $\beta(z_0)$ is reconstructed using $z_1,z_2,z_3,z_4$ and $\beta(z_1)$ using $z_0,z_2,z_3,z_4$, $\beta(z_2)$ using $z_0,z_1,z_3,z_4$ and $\beta(z_3)$ using $z_0,z_1,z_2,z_4$. In this example, if $\beta(z_0)$, $\beta(z_2)$ and $\beta(z_3)$ are detected to be distorted (not codewords of the robust code $C$), then obviously $z_1$ is cheating. If all four secrets are distorted, then $z_4$ is cheating.

We note that the $(A \times N)$ matrix $M$ provide for identification of up to $m$ cheaters if and only if componentwise $OR$ of up to $m$ columns of $M$ are all different and not equal to the column of all $zero's$. These matrices are known as $m-$disjunct or $m-$separable matrix and are closely related to superimposed codes. Methods of construction of these matrices and bounds on the minimal number of rows in these matrices can be found in [23,24,25].

The minimal number $N$ to identify $m$ cheaters $(m < l)$ is $N = l+m$. In this case, number of computed secret is $A = \binom{l+m}{l} - 1$. To compute these secrets we use all subsets of $l$ out of $N$ participants except for one subset, $\{z_0, \cdots, z_{l-1}\}$ as an example.

*Example 5.* For $m = 2, l = 4, N = 6$, we select the following $(14 \times 6)$ matrix $M_2$ for subsets selection. If, for example, all computed secrets are distorted except for the row 7 for which $z_0, z_2, z_4$ are used to compute a secret, then $z_0$ and $z_3$ are cheating.

$$M_2 = \begin{vmatrix} 0&0&1&1&1&1 \\ 0&1&0&1&1&1 \\ 0&1&1&0&1&1 \\ 0&1&1&1&0&1 \\ 0&1&1&1&1&0 \\ 1&0&0&1&1&1 \\ 1&0&1&0&1&1 \\ 1&0&1&1&0&1 \\ 1&0&1&1&1&0 \\ 1&1&0&0&1&1 \\ 1&1&0&1&0&1 \\ 1&1&0&1&1&0 \\ 1&1&1&0&0&1 \\ 1&1&1&0&1&0 \end{vmatrix}.$$

We note that there is a trade off between the number of participants $N$ and the number of secrets $A$ which should be computed for identification of cheaters. Minimizing $A$ results in a minimal complexity of hardware for cheater identification.

*Example 6.* Here are are the matrix $M_3$ and $M_4$ minimizing $A$ for $l = 4, m = 1$ and $l = 4, m = 2$, correspondingly

$$M_3 = \begin{vmatrix} 0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1 \end{vmatrix},$$

$$M_4 = \begin{vmatrix} 0&0&0&0&0&0&1&1&1&1&0&0 \\ 0&0&0&1&1&1&0&0&0&1&0&0 \\ 1&1&1&0&0&0&0&0&0&1&0&0 \\ 0&0&1&0&0&1&0&0&1&0&1&0 \\ 0&1&0&0&1&0&0&1&0&0&1&0 \\ 1&0&0&1&0&0&1&0&0&0&1&0 \\ 0&1&0&1&0&0&0&0&1&0&0&1 \\ 0&0&1&0&1&0&1&0&0&0&0&1 \\ 1&0&0&0&0&1&0&1&0&0&0&1 \end{vmatrix}.$$

For $m = 1, l = 4$, number of participants $N$ for $M_3$ is 7 ($N = 5$ for $M_1$), but number of secrets $A = 3$ is smaller than the one for $M_1$. Comparing $M_2$ and $M_4$ for $l = 4, m = 2$, we can see that $N$ is increasing from 6 to 12 but $A$ is decreasing from 14 to 9. For the case $m = 1$, the minimal number of secrets for cheater identification is $A = \ulcorner log_2 l \urcorner$ and $N = 2l - 1$.

## 5 Simulation and Synthesis results of the proposed schemes

In this section, we will provide the synthesis results for the proposed schemes and the unprotected Shamir's secret sharing architecture to compare the required timing and area redundancy. Then we will run the cheating and fault injection simulation to verify the security level of the proposed architectures.

We note that in this paper, the generator polynomial we use for $GF(2^{32})$ is $f(x) = x^{32}+x^7+x^3+x^2+1$, the generator polynomial used for $GF(2^{96})$ is $f(x) = x^{96} + x^6 + x^5 + x^3 + x^2 + x + 1$, and the generator polynomial used for $GF(2^{128})$ is $f(x) = x^{128}+x^7+x^2+x+1$. All the simulation and synthesis are based on the architectures with parameters $l = 3$, $k = 96$ and $r = 32$.

### 5.1 Synthesis results for the proposed architectures

The proposed secure secret sharing architecture is composed of two parts, the secret reconstruction module and the error checking module. These two modules have been modeled in Verilog and synthesized in Cadence Encounter RTL Compiler with the Nangate 45nm Opencell library version v2009_07. The designs were placed and routed using Cadence Encounter. The latency, the area overhead of the proposed schemes were estimated using Concurrent Current Source (CCS) model under typical operation condition assuming a supply voltage of 1.1V and a temperature of 25 Celsius degree. The synthesis results for different secret sharing schemes are shown in Table 1.

**Table 1.** Resource requirement for the proposed schemes

| | Area ($um^2$) | Latency ($ns$) | Power ($mW$) |
|---|---|---|---|
| Original Secret sharing (96 bits) [I] | 66,217.0 | 1.227 | 77.61 |
| AMD based secret sharing [II] | 112,591.7 | 1.336 | 145.7 |
| Robust based secret sharing [III] | 108,920.4 | 1.433 | 147.3 |

[I] Original secret sharing architecture without error checking module, the size of the secret is 96

[II] Secret sharing module based on AMD code, the size of the information bits of the secret is 96 and redundancy is 32 bits

[III] Secret sharing module based on robust code, the size of the information bits of the secret is 96 and redundancy is 32 bits

Besides the resource overhead listed in Table 1, we note that the Robust and AMD modules also need extra clock cycles to finish the error checking operation. For the original secret sharing module, the clock cycles needed to reconstruct the secret is $13+2*96$, in which 13 clock cycles are used for the control logic and 96 is for the inverter in $GF(2^{96})$. For the proposed architecture, the secret reconstruction module needs $13 + 2 * 128$ clock cycles and the error checking module based on AMD codes needs 7 extra clock cycles while the one based on Robust codes needs 5 extra clock cycles.

From Table 1, we can see that for a 96 bits secret, the proposed AMD based architecture (composed of the 128 bits secret reconstruction and the AMD module) will need 70.3% more area than the original architecture (the 96 bits secret reconstruction module). The proposed robust code based architecture will need 64.5% more area than the original architecture.

For this architecture, we use serial inverters and fully parallel multipliers in finite field [26,27,28] for all the computations. Synthesis results show that while the secret reconstruction module takes $123,025$ gates, the multiplier in $GF(2^{128})$ takes $86,039$ gates. The multiplier in $GF(2^{32})$ takes $4,431$ gates while the AMD module takes $8,472$ gates and the robust module takes $6,291$ gates. So, in our constructions, the multipliers take most of the resource of the whole architecture.

### 5.2 The simulation results for cheating

In this section, two simulations to verify the security level of the proposed schemes will be discussed. According to the discussion in Section 3, the robust code based architecture should be able to provide a high security level under *Type 1* cheating model, and the AMD code based architecture will provide high security levels under *Type 2-4* cheating models.

**Robust system** In the proposed robust code based scheme, the construction polynomial we use is $f(y) = y_0y_1 + y_2^3$ as described in Section 3.2. Under *Type 1* cheating model, the number of attackers is less than $l$ and the attackers have no knowledge of the secret $S_{l-1}$, which means they have no knowledge of the information bits $(y_0, y_1, y_2)$.

For the simulation, we randomly choose 800 sets of IDs and errors, then run 20 million trials of cheating for each set. The simulation is as following:

1. Randomly generate a group of IDs and shares, randomly generate random errors to be injected by 1 to $l-1$ cheaters, the *counter* increases by 1;
2. Randomly generate 20 million shares and for each secret generate shares for the IDs selected in *Step 1*. Inject errors in *Step 1* into the shares and reconstruct the distorted secret, then check how many of the distorted secrets can pass the error checking module;
3. Go back to *Step 1* until *counter* arrives 800.

The result shows that out of these 16 billion trials, there were 5 errors undetected by the error checking module based on robust code. This result is consistent with the upper bound of robust code, which is the probability of missing an error [15,16,17].

**AMD system** For the AMD based architecture, the basic assumption is that $y$ is known to the cheaters while the random numbers $x_0, x_1$ are unknown. Cheaters attempting to cheat the device will have to guess the values of $x_0, x_1$.

Similar to the simulation for the architecture based on robust codes, we ran 800 sets of new secrets and 20 million trials for each set. For each set, we perform the following actions:

1. Randomly select a new polynomial for Shamir's secret sharing with a new random secret;
2. Randomly generate $l = 3$ IDs for $l$ participants and generate their shares using the new polynomial;
3. Randomly generate up to $l = 3$ random errors for the shareholders and inject the errors into their shares.

For each trial, $x_0$ and $x_1$ are randomly generated for the AMD module. The result shows that there are 7 undetected errors out of 16 billion trials.

The cheating simulation results of the two proposed architectures are shown as in Table 2.

From the above cheating simulation results we can see that the proposed schemes are resistant to cheating and have very high security level.

### 5.3 Fault injection at gate level attacks simulation

Besides cheating, the attackers may also directly inject faults into the hardware thus to distort the output [12,13,14]. So the proposed architecture should also be resistant to fault injection attacks. Advanced attackers may have knowledge of the architecture of the target platform and they can use such knowledge

**Table 2.** Results of cheating simulation

| | Cheating | Undetected errors |
|---|---|---|
| AMD<br>based architecture | 16 *billion* | 7 |
| Robust<br>based architecture | 16 *billion* | 5 |

to inject faults into the target and bypass the protection of the proposed schemes. In this section, we will describe the fault injection simulation results of the proposed architectures to verify the security level of the proposed architecture under fault injection attacks.

In fault injection simulation, we inject faults into the multiplier in $GF(2^{32})$ which is the major component of error checking module and the multiplier in $GF(2^{128})$ which is the major component in the secret reconstruction module.

**fault injection into robust code based architecture** We randomly select $1-5$ gates in the hardware and flip the output of these gates. The simulation composed of several steps:

1. Randomly select $1-5$ gates and flip the output of these gates;
2. Randomly choose a legal set of input/output parameters $(z_0, z_1, z_2, s_0, s_2, s_3, y, R)$;
3. Input the legal set into the system with the faults generated in step 1);

If the *error* signal is set, then it means the injected fault is detected; if the *error* signal is not set and the output is the original secret $y$, then it means the fault is masked; if the *error* signal is not set and the output is not the original secret $y$, then it means the attacker successfully generated an undetectable error. For the architecture based on robust code, we run 50.3 million rounds of the above simulation and the result shows that only one error is undetected.

**fault injection into AMD based architecture** For AMD code, the simulation is almost the same as the one for robust code except for each legal set is $(z_0, z_1, z_2, s_0, s_2, s_3, x_0, x_1, y, R)$ which contains two more random numbers $x_0, x_1 \in GF(2^{32})$.

For the AMD based architecture, we run 52.3 million rounds of the fault injection simulation, and the AMD module detected all the errors.

The fault injection simulation results of the AMD and robust based schemes are shown as in Table 3.

**Table 3.** Fault injection simulation results

| | Injected faults | Undetected errors |
|---|---|---|
| AMD<br>based architecture | 52.3 *million* | 0 |
| Robust<br>based architecture | 50.3 *million* | 1 |

Table 3 shows that the proposed schemes based on robust and AMD codes can also provide high security level under fault injection attacks. The proposed schemes can detect most of the injected errors with very low probability of missing an error and thus improve the security level of the system significantly.

## 6 conclusion

In this paper, we described a kind of secure and reliable secret sharing architectures which are resistant to cheating and fault injection attacks based on robust and AMD codes. We analyzed the security level of the the proposed schemes under different cheating models and also ran fault injection simulation to test the security level of the proposed schemes. Results show that the robust code based architecture is secure under *Type 1* cheating model. The AMD code based architecture is secure even under *Type 2-4* cheating models. Both architectures can detect the injected faults with very high probability.

At the same time, we also implement the proposed schemes in Verilog and ran synthesis to estimate the overhead. Results show that the proposed architectures need slightly larger overhead than the unprotected architecture.

**Acknowledgment**: The work of the fourth author is sponsored by the NSF grant CNS 1012910. This work is equally contributed by the first two authors.

## References

1. G. R. Blakley, “Safeguarding Cryptographic Keys,” in *Proceedings of the 1979 AFIPS National Computer Conference*, vol. 48, Jun. 1979, pp. 313–317.
2. A. Shamir, “How to share a secret,” *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
3. M. Tompa and H. Woll, “How to share a secret with cheaters,” in *Proceedings on Advances in cryptology—CRYPTO ’86*. London, UK, UK: Springer-Verlag, 1987, pp. 261–265.
4. L. Harn and C. Lin, “Detection and identification of cheaters in (t, n) secret sharing scheme,” *Des. Codes Cryptography*, vol. 52, no. 1, pp. 15–24, Jul. 2009.
5. Y.-X. Liu, L. Harn, C.-N. Yang, and Y.-Q. Zhang, “Efficient (n, t, n) secret sharing schemes,” *Journal of Systems and Software*, vol. 85, no. 6, pp. 1325 – 1332, 2012.
6. T.-C. Wu and T.-S. Wu, “Cheating detection and cheater identification in secret sharing schemes,” *Computers and Digital Techniques, IEE Proceedings -*, vol. 142, no. 5, pp. 367–369, 1995.
7. H.-Y. Lin and L. Harn, “A generalized secret sharing scheme with cheater detection,” in *Proceedings of the International Conference on the Theory and Applications of Cryptology: Advances in Cryptology*, ser. ASIACRYPT ’91. London, UK, UK: Springer-Verlag, 1993, pp. 149–158.
8. D. Pasaila, V. Alexa, and S. Iftene, “Cheating detection and cheater identification in crt-based secret sharing schemes.” *IACR Cryptology ePrint Archive*, vol. 2009, p. 426, 2009.
9. T. Araki, “Efficient (k, n) threshold secret sharing schemes secure against cheating from n - 1 cheaters,” in *Proceedings of the 12th Australasian conference on Information security and privacy*, ser. ACISP’07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 133–142.
10. C. Blundo, A. D. Santis, L. Gargano, A. De, S. L. Gargano, and U. Vaccaro, “Secret sharing schemes with veto capabilities,” in *in: Proceedings of French-Israeli Workshop in Algebraic Coding*. Springer-Verlag, 1994, pp. 82–89.
11. R. J. McEliece and D. V. Sarwate, “On sharing secrets and reed-solomon codes,” *Commun. ACM*, vol. 24, no. 9, pp. 583–584, Sep. 1981. [Online]. Available: http://doi.acm.org/10.1145/358746.358762
12. S. Skorobogatov, “Optical fault masking attacks,” in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2010 Workshop on*, 2010, pp. 23–29.
13. ——, “Local heating attacks on flash memory devices,” in *Hardware-Oriented Security and Trust, 2009. HOST ’09. IEEE International Workshop on*, 2009, pp. 1–6.
14. S. P. Skorobogatov, “Semi-invasive attacks – A new approach to hardware security analysis,” University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-630, Apr. 2005.
15. M. Karpovsky, K. Kulikowski, and A. Taubin, “Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard,” in *Dependable Systems and Networks, 2004 International Conference on*, june-1 july 2004, pp. 93 – 101.
16. M. Karpovsky and A. Taubin, “New class of nonlinear systematic error detecting codes,” *Information Theory, IEEE Transactions on*, vol. 50, no. 8, pp. 1818 – 1819, aug. 2004.
17. K. Kulikowski, Z. Wang, and M. Karpovsky, “Comparative analysis of robust fault attack resistant architectures for public and private cryptosystems,” in *Fault Diagnosis and Tolerance in Cryptography, 2008. FDTC ’08. 5th Workshop on*, aug. 2008, pp. 41 –50.
18. Z. Wang, M. Karpovsky, and A. Joshi, “Nonlinear multi-error correction codes for reliable mlc nand flash memories,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, no. 7, pp. 1221 –1234, july 2012.
19. Z. Wang and M. Karpovsky, “Algebraic manipulation detection codes and their applications for design of secure cryptographic devices,” in *On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International*, july 2011, pp. 234 –239.

20. ——, “Reliable and secure memories based on algebraic manipulation correction codes,” in *On-Line Testing Symposium (IOLTS), 2012 IEEE 18th International*, june 2012, pp. 146 –149.
21. R. Cramer, Y. Dodis, S. Fehr, C. Padr, and D. Wichs, “Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors,” in *Advances in Cryptology C EUROCRYPT 2008*, ser. Lecture Notes in Computer Science, N. Smart, Ed. Springer Berlin / Heidelberg, 2008, vol. 4965, pp. 471–488.
22. L. Pei, W. Zhen, and K. Mark, “Secure nand flash architecture resilient to strong fault-injection attacks using algebraic manipulation detection code,” in *The 2013 International Conference on Security and Management*, 2013.
23. E. Porat and A. Rothschild, “Explicit nonadaptive combinatorial group testing schemes,” *IEEE Trans. Inf. Theor.*, vol. 57, no. 12, pp. 7982–7989, Dec. 2011. [Online]. Available: http://dx.doi.org/10.1109/TIT.2011.2163296
24. V. V. R. A. G. D’yachkov, “Bounds on the length of disjunctive codes,” in *Problems Inform. Transmission*, vol. 18. Springer-Verlag, 1982, pp. 7–13.
25. Z. Füredi, “On r-cover-free families,” *J. Comb. Theory Ser. A*, vol. 73, no. 1, pp. 172–173, Jan. 1996. [Online]. Available: http://dx.doi.org/10.1006/jcta.1996.0012
26. N. Jachimiec, N. Iliev, and J. Stine, “Strategies for vlsi implementations of finite field inversion algorithms,” in *Circuits and Systems, 2005. 48th Midwest Symposium on*, 2005, pp. 1589–1592 Vol. 2.
27. Z. Yan and D. Sarwate, “Modified euclidean algorithm for finite field inversions,” in *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, 2002, pp. 203–.
28. J. Fan and I. Verbauwhede, “Extended abstract: Unified digit-serial multiplier/inverter in finite field gf(2m),” in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, 2008, pp. 72–75.