RASSS: A Perfidy-Aware Protocol for Designing Trustworthy Distributed Systems

Lake Bu, Hien D. Nguyen, Michel A. Kinsy, *Member, IEEE*
Adaptive and Secure Computing Systems Laboratory
Boston University, Boston, USA

***Abstract*—Robust Adaptive Secure Secret Sharing (RASSS) is a protocol for reconstructing secrets and information in distributed computing systems even in the presence of a large number of untrusted participants. Since the original Shamir's Secret Sharing scheme, there have been efforts to secure the technique against dishonest shareholders. Early on, researchers determined that the Reed-Solomon encoding property of the Shamir's share distribution equation and its decoding algorithm could tolerate cheaters up to one third of the total shareholders. However, if the number of cheaters grows beyond the error correcting capability (distance) of the Reed-Solomon codes, the reconstruction of the secret is hindered. Untrusted participants or cheaters could hide in the decoding procedure, or even frame up the honest parties. In this paper, we solve this challenge and propose a secure protocol that is no longer constrained by the limitations of the Reed-Solomon codes. As long as there are a minimum number of honest shareholders, the RASSS protocol is able to identify the cheaters and retrieve the correct secret or information in a distributed system with a probability close to 1 with less than 60% of hardware overhead. Furthermore, the adaptive nature of the protocol enables considerable hardware and timing resource savings and makes RASSS highly practical.**

I. Introduction

In many applications and systems, secret sharing techniques are deployed when a piece of confidential data cannot be entrusted to a single person. The general concept consists of taking that piece of data, i.e., the secret, and sharing it among multiple holders, each with unique ID, in a manner that allows the reconstruction of the shared secret using inputs from only a subset of the shareholders. The minimum size of any subset to reconstruct the secret is called "threshold". Below the threshold, the secret is information theoretically safe and cannot be retrieved.

Practical secret sharing techniques are required in many real world applications. For example, the DNS Security (DNSSEC) [1] which ensures the DNS (Domain Name System) servers to connect the users and their Internet destinations (URLs and IPs) in a secure and verified manner, has its root key split and shared among seven holders. In the case of an attack, if five or more of the holders are in the same U.S. base, then they can reconstruct the root key using their shares and restore the Internet connections. Another application is in Hardware Security Module (HSM) based systems. HSMs are widely used in bank card payment systems. Some HSMs [2] are produced and distributed by certification authorities (CAs) and registration authorities (RAs) to generate and share important secret keys under Public Key Infrastructure (PKI). These HSMs also require implementation of a multi-part user authentication scheme, namely threshold secret sharing.

Due to their distributed nature, secret sharing techniques are susceptible to a number of attacks, like, man-in-the-middle attacks and share manipulations, i.e., cheating. These attacks, resulting in share distortions, may lead to the retrieval of a wrong secret. Although, there are many secure secret sharing schemes, they are often limited in their cheater tolerance. Generally, the number of cheaters exceeds their fault tolerance or error correction capabilities. Therefore, to improve the robustness of secret sharing in distributed systems, we propose a new protocol tolerating a large number of untrusted and colluding participants, called *Robust Adaptive Secure Secret Sharing* (RASSS). The contributions of this work are:

1) The protocol tolerates beyond the previously established $t < n/3$ cheater tolerance bound, where t denotes the number of cheaters and n the number of parties engaged in the computation. The new protocol is able to reconstruct secrets as long as there exists a minimum threshold number of honest parties, where the classic protocol using Reed-Solomon decoder is unable to either identify the cheaters or retrieve the correct secret;
2) The protocol has a higher level of security. It is able to detect cheating conducts and identify the cheaters even when there is sophisticated collusion among them. In contrast, under this situation the classic protocol will be misled and will retrieve an erroneous secret and/or mislabel the honest parties;
3) The new scheme is adaptive, which allows for efficient implementation with low computation complexity on average. In our design and analysis, RASSS shows a hardware overhead of only 60% over the classic protocol.

The rest of the paper is organized as follows. Sections II introduces the original secret sharing scheme and its secure protocol to tolerate up to $t < n/3$ cheaters. Section III explains the vulnerability of this classic protocol when $t \geq n/3$. Section IV proposes the new RASSS protocol to overcome the vulnerability. Section V is on the analysis of the security level and overhead of the RASSS protocol.

II. The Original Shamir's Secret Sharing Scheme and Its Classic Secure Protocol

The following notations are used to describe and evaluate the original Shamir's secret sharing scheme, the classic and proposed secure protocols:

- b: the number of bits in a vector variable;
- S: the original secret;
- x_i: the ID number of the i^{th} shareholder;
- D_i: the share of the i^{th} shareholder;
- k: the minimum number (threshold) of shareholders needed to reconstruct a secret;
- t: the number of cheaters;
- n: the total number of shareholders in computation;
- T: the number of tests needed to identify the cheaters and honest shareholders;
- AMD: the algebraic manipulation detection codes;

978-1-5386-0362-8/17/$31.00 ©2017 IEEE

- E: the encoded secret by AMD codes;
- P_{mask}: the error masking probability of AMD codes;
- RS: the Reed-Solomon codes;
- d: the distance of RS codes where $d = n - k + 1$ which tolerates (or corrects) up to $\frac{d-1}{2}$ errors;
- $\oplus$: the addition operator in finite fields.

A. The Original Shamir's Secret Sharing

The concept of k-threshold secret sharing was first introduced by Shamir [3] in 1979. For the sake of information theoretic security, all elements and operations are supposed to work under Galois finite field (GF) arithmetic where the field size should be a prime or power of prime. To share a secret S, a polynomial is used to distribute the shares where the secret S serves as the leading coefficient. The shares are the evaluations of the polynomial by each shareholder's ID x_i:

$$D_i = c_0 \oplus c_1 x_i \oplus c_2 x_i^2 \oplus \cdots \oplus S x_i^{k-1}. \tag{1}$$

Usually the ID number is publicly known to everyone while the shares are kept private by shareholders. With any subset of at least k shareholders' IDs and shares, one can use the Lagrange interpolation formula to reconstruct the secret:

$$S = \bigoplus_{i=0}^{k-1} \frac{D_i}{\prod_{j=0, j\neq i}^{k-1} (x_i \oplus x_j)}. \tag{2}$$

Such a construction is $(k-1)$-private. This means it needs at least k shareholders to reconstruct the secret and so any $(k-1)$ shareholders have no knowledge of the secret.

B. The Classic Secure Protocol for Share Verification

After the invention of Shamir's secret sharing, it was noticed that if any number of the shareholders participating in the secret reconstruction apply an active attack by changing their shares, the retrieved secret will be distorted. Therefore Cramer *et. al.* [4] have proposed an Algebraic Manipulation Detection (AMD) code to detect any modification of secrets with a probability close to 1. Karpovsky *et. al.* [5] later generalized this code with a flexible construction. On the other hand, researchers [6], [7], [8] have proposed approaches to verify the validity of shares with a probability of 1. The common feature in the latter approaches is that, if the shares can be encoded to a codeword of a certain error control code (ECC), then the codeword's symbols (shares) can be verified and corrected within the ECC's capability.

Particularly, the share distribution [Eq. 1] is inherently equivalent to the non-systematic encoding equation of the well-known Reed-Solomon (RS) ECC codes. RS codes are maximum distance separable (MDS) codes which meet the Singleton bound with equality. With such a distribution equation, an (n, k, d) Reed-Solomon codeword $(D_0, D_1, \cdots D_{n-1})$ is encoded with n symbols (shares) in total, k information symbols, and distance $d = n-k+1$ which corrects up to $\frac{d-1}{2}$ (or $\frac{n-k}{2}$) erroneous symbols with algorithms in [9], [10].

In the secret sharing language, with n shareholders' IDs and shares, we are able to tolerate up to $t \leq \frac{n-k}{2}$ shares maliciously modified by cheaters. Theoretically speaking, the error correction capability of RS codes can tolerate up to $t < n/2$ cheaters if $n \gg k$. However, oftentimes an assumption is made that there should be $t < k$ cheaters such that a group of all cheaters have no access to the secret [11]. Then we have:

$$t < n/3. \tag{3}$$

If n instead of k shareholders are involved in the share error correction by RS decoders, then the correctness of the retrieved secret is ensured when [Eq. 3] holds. Consequently, the secure secret sharing is both t-private and t-resilient, that up to t shareholders cannot reconstruct the secret, and up to t cheaters cannot affect the correctness of the secret [12].

III. Vulnerabilities of the Classic Secure Protocol

The essence of the classic RS-based secure scheme is to encode the shares into a codeword, whose validity can be verified by the RS decoding algorithm. Although RS codes are known for their strong error correction, their encoding procedure is linear and susceptible to cheating exploits.

We assume a strong attack model, that the cheaters can change their shares to any value, and they are all colluding. With t beyond the error correction capability of the chosen RS code, the cheaters, collusively, can breach the safeguards of the protocol. To illustrate what types of attacks they can implement, we will use the relationship between t and d (the RS code's distance) to describe the increasing vulnerability in the protocol when t increases.

A. Making the Secret Unaccessible

If the number of cheaters satisfy $\frac{d-1}{2} < t < d$, although the RS decoder can still raise an alarm for cheating, it is unable to retrieve the secret or identify the cheaters.

B. Turning Off the Alarm

If the number of cheaters satisfies $d \leq t \leq n$, they will be able to manipulate the entire system. For instance the cheaters can pick another share distribution polynomial different from [Eq. 1] with random coefficients b_i and another secret S':

$$D_i' = b_0 \oplus b_1 x_i \oplus b_2 x_i^2 \oplus \cdots \oplus S' x^{k-1} \tag{4}$$

The new shares D_i' of the cheaters will be the evaluation of [Eq. 4] by the same IDs x_i. When $t \geq d$, the cheaters' shares will form a new legal RS codeword which will never be detected by the RS decoder. The secret reconstruction will then produce the secret S' that the cheaters have selected.

Example III.1. A secret sharing system has a secret $S = 111$ in the $GF(2^3)$ finite field. It requires $k = 2$ shareholders to reconstruct the secret every time. The following share distribution polynomial is used to generate the shares:

$$D_i = c_0 \oplus S x_i = 010 \oplus 111 x_i.$$

The protocol is designed in such a way that up to 1 cheater can be tolerated. Therefore, in the secret reconstruction stage there will be $n = 3t + 1 = 4$ shareholders involved. Suppose that in the secret reconstruction, shareholders with IDs $x_0 = 001, x_1 = 010, x_2 = 011, x_3 = 100$ are involved. And the shares distributed to them are $D_0 = 101, D_1 = 111, D_2 = 010, D3 = 001$. These 4 shares form a legal RS codeword $v = (101, 111, 010, 001)$ with distance $d = 3$ and it can correct up to 1 error.

Now all 4 of them are cheating collusively, and they have selected their own secret $S' = 100$ and a different share distribution polynomial:

$$D_i' = b_0 \oplus S' x_i = 001 \oplus 100 x_i.$$

Thus their shares will be maliciously changed to $D_0 = 101, D_1 = 010, D_2 = 110, D3 = 111$, which is also a legal

codeword $v' = (101, 010, 110, 111)$ of a $(n, k, d) = (4, 2, 3)$ RS code. This codeword will unfortunately be considered as a valid codeword by the RS decoding algorithm [10] and there will be no cheating alarm. As a result, the fake secret $S' = 100$ is retrieved by those shares under [Eq. 2]. During the entire procedure the cheating will not be detected. □

C. Framing Up the Honest Shareholders

Another vulnerability that cheaters can exploit when $d \leq t \leq n$ is to frame up the honest shareholders, so that the decoder treats the honest parties as "cheaters" and cheaters as "honest shareholders". If t is large enough that the number of honest shareholders is $n - t \leq \frac{d-1}{2}$, then the honest shareholders are within the RS decoder's error correction capability. Since all cheaters' shares are generated by the same forged secret sharing polynomial, the honest minority will be treated as cheaters and "corrected". The cheaters' fake secret will be regarded as the valid secret as the result of [Eq. 2].

Example III.2. Suppose that we have the same secret sharing system as in Example III.1. Let us have three shareholders $\{x_0 = 001, x_1 = 010, x_2 = 011\}$ as cheaters, and shareholder $x_3 = 100$ is an honest participant. The codeword for the shares submitted to the RS decoder will be $v' = (101, 010, 110, 001)$. v' will be decoded as $(101, 010, 110, 111)$ which is the cheaters' codeword since $d \leq t$. Shareholder $x_3 = 100$ will be labeled as a "cheater". Consequently, the forged secret $S' = 100$ (as in Example III.1) will be retrieved. □

IV. The Robust Adaptive Secure Secret Sharing (RASSS) Protocol

We have shown that (1) the RS-based protocol has a limited cheater correction capability with probability of 1, and (2) the AMD codes provides strong detection of any modification to the secret with a probability close to 1. Inspired by these properties, we propose a new robust adaptive secure secret sharing (RASSS) protocol for secret sharing using both techniques for cheater identification and correction. The advantages of the new protocol are:

1) When $0 \leq t < n/3$ (or $0 \leq t \leq \frac{d-1}{2}$): the AMD-based protocol detects the cheating and the proposed protocol (a) corrects all the cheaters' shares and (b) retrieves the correct secret with a probability of 1, same as in the RS-based approach;
2) When $n/3 \leq t \leq n - k$ (or $\frac{d-1}{2} < t < d$): the proposed protocol is able to identify all the cheaters and retrieve the correct secret with a probability close to 1. Both the RS and AMD-based protocols only detect the cheating, but they are unable to either identify the cheaters or retrieve the correct secret;
3) When $n - k < t \leq n$ (or $d \leq t \leq n$) and there are not enough honest shareholders to retrieve the secret: the proposed protocol detects cheating with a probability close to 1, same as in the AMD-based protocol. With additional resources, the proposed approach is able to identify cheaters and retrieve the secret. Whereas, the RS-based protocol will retrieve a fake secret and mislabel the honest shareholders as cheaters.

The following subsections are organized in the order of: overview of the proposed 4-stage protocol, detailed introduction of the submodules of the four stages, and a numeric illustrative example of the protocol's mechanics.

A. Overview of the RASSS Protocol

The RASSS protocol has four conditional branches to switch among the stages.

Stage 1: Secret Encoding and Share Distribution
In the first stage the protocol will encode the secret S with the Algebraic Manipulation Detection (AMD) encoder. The encoded secret E is then distributed using equation [Eq. 1].

Stage 2: Secret Reconstruction and Verification
A set of k shareholders will participate in the secret reconstruction using [Eq. 2]. The retrieved secret will be decoded and verified by the AMD decoder module. If the decoder claims validity of the secret, then it is considered a successful secret reconstruction with no cheating involved. If not, the protocol calls for Stage 3.

Stage 3: Share Error Correction
This stage uses the Reed-Solomon error correction module in the classic protocol. Here, $n = 3t + 1$ shareholders will be invited to participate in the protocol, where t is the number of estimated cheaters defined by the system. The RS decoder will try to correct the shares and then send them back to the secret reconstruction and verification modules. If it passes both the share correction (by RS decoder module) and secret verification (by AMD decoder module), then the secret reconstruction is successful. If either module fails then the protocol ascends to its fourth stage, indicating that the actual number of cheaters is greater than $n/3$.

Stage 4: Group Testing
This stage will generate a group testing pattern in the form of a binary matrix M of size $T \times n$. As long as there are at least k honest shareholders, this stage is always capable of retrieving the correct secret while identifying up to $n/3 \leq t \leq n - k$ cheaters within T tests. If $n - k < t$ and there are less than k honest parties, this stage is still capable of detecting cheating. The protocol can be extended to include an invitation module. The purpose of such a module is to pull in the operation additional parties or system nodes to increase the number of potential honest participants. The work flow of RASSS is shown in the figure below.

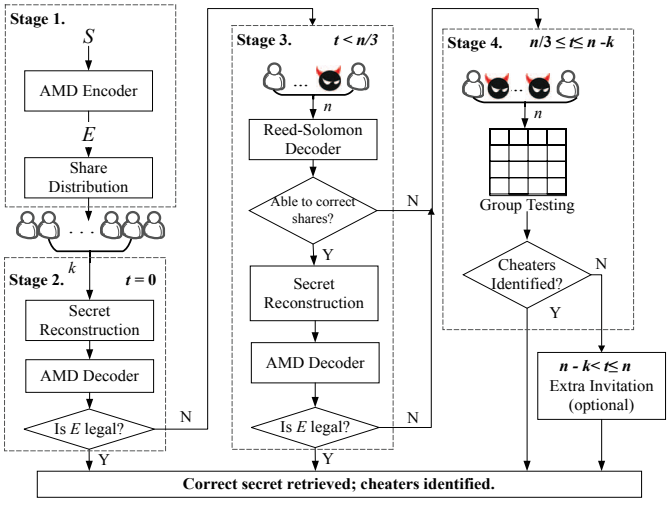

Fig. 1: **Stage 1** and **2** are sufficient if the number of cheaters $t = 0$. If cheating is detected by **Stage 2**, then **Stage 3** with RS decoder is called under the assumption of $t < n/3$. If **Stage 3** fails then **Stage 4** with group testing is able to identify $n/3 \leq t \leq n-k$ cheaters and retrieve the correct secret. If t is even beyond this scale, an additional invitation module can be introduced to resolve the issue.

The graduate, stage-based, adaptive nature of the RASSS protocol ensures that a higher security stage with greater computational cost is activated only if a conditional branch determines that the current stage is inadequate. Under this approach, the execution time complexity and resource utilization are application driven and on average (most common case) minimum. The stages' submodules are introduced in the following subsections.

B. Secret Encoding module (AMD Encoder)

The linearity of the RS encoding and its vulnerabilities (cf. III) enable the attackers to forge legal shares with the knowledge of (n, k, d). In the RASSS protocol, we encode the secret with security-oriented AMD codes [5] so that forged shares will not result in valid secrets. Under the protocol, encoded secrets are distributed instead of original "raw" secrets. This way, the authenticity of retrieved secrets can also be verified.

Definition IV.1. Let $R = (R_1, R_2, \cdots, R_m)$, where $R_i \in GF(2^b)$ is a randomly generated b-bit vector. An h^{th} order Generalized Reed-Muller code (GRM) [13] with m variables consists of all codewords $(f(0), f(1), \cdots, f(2^{bm}-1))$, where $f(R)$ is a polynomial of $R = (R_1, R_2, \cdots, R_m)$ of degree up to h. Let

$$A(R) = \begin{cases} \bigoplus_{i=1}^{m} R_i^{h+2}, & \text{if } h \text{ is odd;} \\ \bigoplus_{i=2}^{m-1} R_1 R_i^{h+1}, & \text{if } h \text{ is even and } m > 1; \end{cases}$$

where $\bigoplus$ is the accumulated sum in $GF(2^b)$. Let

$$B(R, y) = \bigoplus_{1 \le j_1 + j_2 + \cdots + j_1 \le h+1} y_{j_1, j_2, \cdots, j_m} \prod_{i=1}^{m} R_i^{j_i},$$

where $\prod_{i=1}^{m} R_i^{j_i}$ is a monomial of R of a degree between 1 and $h+1$. And $\prod_{i=1}^{m} R_i^{j_i} \notin \triangle B(R, y)$ which is defined by:

$$\begin{cases} \{R_1^{h+1}, R_2^{h+1}, \cdots, R_m^{h+1}\}, \text{if } h \text{ is odd;} \\ \{R_2^{h+1}, R_1 R_2^{h}, \cdots, R_1 R_m^{h}\}, \text{if } h \text{ is even and } m > 1. \end{cases}$$

Let $f(R, y) = A(R) \oplus B(R, y)$, then a generalized AMD codeword is composed of the vectors $(y, R, f(R, y))$, where y is the information portion, R the random vector, and $f(R, y)$ the redundancy portion [5]. ■

Remark IV.1. If the attack involves an error $e_y \neq 0$ on the information y, which is the major purpose of almost all attacks, then in $f(R, y)$ the term $A(R)$ can be omitted [14]. ■

In the RASSS protocol, by a randomly generated vector R and the AMD encoding equation for $f(R, y)$, the original secret S is encoded into:

$$E = (S, f(R, S)). \tag{5}$$

We call this newly generated E the encoded secret. It will be shared using [Eq. 1] to all shareholders instead of the original secret S. And the random vector R will be sent to the secret decoding module after every secret encoding.

C. Secret Decoding module (AMD Decoder)

The secret reconstruction procedure will retrieve a secret that is probably distorted under the existence of cheaters. If we denote the error caused by cheating as $e = (e_f, e_R, e_S)$, and the distorted secret as $\widetilde{E} = (\widetilde{S}, \widetilde{f(R, S)})$, then the AMD decoder is to check whether the following equation holds:

$$\widetilde{f(R, S)} \stackrel{?}{=} f(\widetilde{R}, \widetilde{S}) \tag{6}$$

where $\widetilde{f(R, S)} = f(R, S) \oplus e_f$, $\widetilde{S} = S \oplus e_S$, $\widetilde{R} = R \oplus e_R$, assuming for the worst case scenario that R is also erroneous.

If [Eq. 6] is not equal, then an error is detected. If $\widetilde{E} \neq E$ but [Eq. 6] still holds the equality, then the error is masked. The security level of the AMD codes is defined by the error masking probability P_{mask} when $e \neq 0$.

By **Remark IV.1**, if $e_S \neq 0$ and so $f(R, S) = B(R, S)$, [Eq. 6] can be written as the error masking equation (EME):

$$B(R, S) \oplus e_f \oplus B(R \oplus e_R, S \oplus e_S) = 0. \tag{7}$$

It is fairly easy to determine that the left side of [Eq. 7] is a non-zero polynomial of R of a degree up to h, and R has at most h solutions out of all 2^b possible values. Any error caused by attacks that makes [Eq. 7] hold, will be masked. Therefore for any given error $e = (e_f, e_R, e_S)$ where $e_S \neq 0$, the security level of AMD codes characterized by the error masking probability P_{mask} can be upper bounded by:

$$\overline{P_{mask}} = \frac{h}{2^b}. \tag{8}$$

It is obvious that as b increases, the error detection probability $(1 - P_{mask})$ grows rapidly close to 1 and the AMD code becomes more secure.

D. Share Error Correction Module (RS Decoder)

Without loss of generality, when Stage 2 detects that the reconstructed secret is invalid, it is reasonable and practical to initially assume that t is not a large number. Therefore, Stage 3 involves $n = 3t + 1$ shareholders and tolerates up to t cheaters with a probability of 1 by the RS decoder. The secret retrieved from the corrected shares will still be verified by the AMD decoder module in the case of the collusive attacks (cf. Section III-B and III-C).

E. Group Testing Module

If Stage 3 fails and the actual number of cheaters $t \geq n/3$, Stage 4 is enabled to identify cheaters and retrieve the secret with a probability close to 1 using secret verification and group testing. Stage 4 uses group testing to tolerate cheaters in the range of $n/3 \leq t \leq n - k$ (or $\frac{d-1}{2} < t < d$). The lower bound is beyond the capability of the classic RS decoder module, and the upper bound is tightly roofed by only k honest shareholders (the minimum number required to retrieve the secret). This means that out of $\binom{n}{k}$ possible subsets of shareholders, there can be as few as 1 subset only to retrieve the correct secret. The test pattern is described in the following construction.

Construction IV.1. For any secret sharing scheme that is $(k-1)$-private, suppose among n shareholders there are t cheaters where $n/3 \leq t \leq n - k$. A test pattern to identify the honest and cheating parties can be constructed as a binary matrix M of size $T \times n$, where T is the number of tests needed. The rows of M consist of all different n-bit vectors with exactly k 1's and so $T = \binom{n}{k}$. Each column of the matrix therefore has $\binom{n-1}{k-1}$ number of 1's. The 1's in each row (test) correspond to the shareholders participating in that particular test. Each test is a two-step procedure:

1) A secret reconstruction using [Eq. 2] to retrieve the secret $\widetilde{E}$ with its specific participants;
2) An AMD decoding using [Eq. 6] over $\widetilde{E}$ to verify the validity of the retrieved secret. The test syndrome is a T-bit binary vector u, where 0's in u indicate the equality of [Eq. 6], and 1's the inequality. ■

Then the cheaters can be identified by the algorithms below.

Algorithm IV.1. For any $(k-1)$-private secret sharing scheme and its corresponding group testing matrix M there are n shareholders participating in the tests indexed by $H = \{0, 1, 2, \cdots, n-1\}$. Among the n shareholders there are t cheaters where $n/3 \leq t \leq n-k$. Let $w = (w_0, w_1, \cdots, w_{n-1})$ be a n-digit vector and $w = u^\top \times M$, where u is the T-bit binary test syndrome and $\times$ is the multiplication of regular arithmetic. The cheaters' indices belong to the set $\{l|\ w_l = \binom{n-1}{k-1}\}$. and the rest of the holders are honest. ■

This test pattern M can be utilized in an adaptive manner to drastically reduce the average number of tests needed.

Algorithm IV.2. For a test pattern M of size $T \times n$ generated by Construction IV.1, $\triangle T$ is the number of tests needed to find the first 0 (equality of [Eq. 6]) in the test syndrome. The k honest holders identified by this test are indexed by $I = \{i_0, i_1, \cdots, i_{k-1}\}$. The system only needs to run at most $n-k$ more tests whose participants are $\{i_0, i_1, \cdots, i_{k-2}, j\}$, where $j \in H \backslash I$. Each test's syndrome indicates holder j as a cheater or not by 1 or 0. The total number of tests needed to identify all holders is then. 0 at most $\triangle T + (n-k)$. ■

F. Extra Invitation Module

If the group testing module in Stage 4 cannot successfully identify the t cheaters in the system, where $n-k < t \leq n$, then the number of honest shareholders is less than k. Unlike the classic one, the RASSS protocol will still raise the cheating alarm based on the AMD decoder module [Eq. 6]. Moreover, the protocol is adaptive enough to be extended to a fourth stage to include an invitation module. This module can pull in the execution additional participants and perform new rounds of group testing. From the hardware prospective, the invitation module can be power-gated and disabled when not in use.

Algorithm IV.3. Let the number of honest shareholders in the current group testing be $\triangle k$ and $0 \leq \triangle k < k$. Suppose the system is able to identify an extra set of k honest shareholders from another group. Then these k honest parties can be combined into the current group with the modified group testing matrix of size $\binom{n+k}{k} \times (n+k)$. With this new test pattern, the $\triangle k + k$ honest shareholders can be identified and the rest will be properly labeled as cheaters. ■

G. An Example of the Proposed RASSS Protocol

Here we present an illustrative example to demonstrate the adaptivity and robustness of the proposed protocol.

Example IV.1. A Shamir's secret sharing scheme is $(k-1)$-private and $k = 3$. The original secret $S \in GF(2^{12})$ where $S = 001111110000 = 0x3F0$. The RS decoder is constructed under the assumption that there are at most 2 cheaters in every secret reconstruction. However, in the actual scenario there are more cheaters than honest shareholders. The RASSS protocol is able to retrieve the correct secret under this grave situation.

Stage 1: Secret Encoding and Share Distribution

The original secret $0x0F0$ is first encoded by the AMD encoding equation [Eq. 5]. Using Definition IV.1. we choose $b = 4$ such that the encoding and decoding are over $GF(2^4)$, $m = 1$ such that the random vector has only one symbol, and $h = 3$ such that S is partitioned into 3 symbols $S = (S_0, S_1, S_2)$ where $S_0 = 0x3, S_1 = 0xF$, and $S_2 = 0x0$. Suppose the random number generator generates $R = 0x6$. The original secret will be encoded to an AMD codeword $E = (S, f(R,S)) = (S, B(R,S))$ by:

$$B(R,S) = S_0 R \oplus S_1 R^2 \oplus S_2 R^3 = 0x1 \Rightarrow E = (0x3F01).$$

Then with the share distribution polynomial:

$$D_i = c_0 \oplus c_1 x_i \oplus E x_i^2$$

where $c_0 = 0xAAAA, c_1 = 0x5555$ are arbitrarily chosen coefficients and $c_0, c_1, E \in GF(2^{16})$, this encoded secret is shared to seven shareholders with IDs and shares $\{x_i : D_i\}$ = $\{1 : 0xC0FE\}$, $\{2 : 0xFC04\}$, $\{3 : 0x9650\}$, $\{4 : 0x0FB4\}$, $\{5 : 0x65E0\}$, $\{6 : 0x591A\}$, $\{7 : 0x334E\}$. However, shareholders $\{3, 4, 6, 7\}$ are cheaters and they have collusively selected another secret $S' = 0xABCD$ and forged another share distribution polynomial:

$$D_i' = 0xBBBC \oplus 0x7777 x_i \oplus 0xABCD x_i^2.$$

By their IDs, their shares are changed to: $\{3 : 0x2686\}$, $\{4 : 0xDBAF\}$, $\{6 : 0x9A2F\}$, $\{7 : 0x4695\}$.

Stage 2: Secret Reconstruction and Verification

Suppose shareholders $\{2, 3, 4\}$ are selected to reconstruct the secret with $\{3, 4\}$ being cheaters. By the secret reconstruction [Eq. 2] the retrieved secret is:

$$\widetilde{E} = 0x5522.$$

The reconstructed secret will be verified by the AMD decoder using [Eq. 6]: $\widetilde{f(R,S)} \stackrel{?}{=} f(\widetilde{R}, \widetilde{S})$, where $\widetilde{f(R,S)} = \widetilde{B(R,S)} = 0x2$, $\widetilde{R} = 0x6$, $\widetilde{S} = (0x5, 0x5, 0x2)$. Through the computation over $GF(2^4)$ we have the following inequality:

$$[\widetilde{B(R,S)} = 0x2] \neq [B(\widetilde{R}, \widetilde{S}) = \widetilde{S_0}\widetilde{R} \oplus \widetilde{S_1}\widetilde{R}^2 \oplus \widetilde{S_2}\widetilde{R}^3 = 0x7].$$

Thus, cheating is detected and Stage 3 will be initiated under the assumption of $t = 2$ cheaters.

Stage 3: Share Error Correction

Under the RS decoder, $n = 3t + 1 = 7$ shareholders will be involved and it can correct up to 2 shares using an $(n, k, d) = (7, 3, 5)$ RS code. However, there is a total number of $t = 4$ cheaters $\{3, 4, 6, 7\}$ which is beyond the capability of this RS decoder. Therefore, the protocol moves in its fourth stage.

Stage 4: Group Testing

This stage is designed under the assumption that among all the shareholders from Stage 3, only $k = 3$ are honest. The group testing matrix M of size $T \times n$ can be constructed with Construction IV.1, where $T = \binom{n}{k} = 35, n = 7$. To save space M is listed in its transposed form $M^\top$:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0
4	0	1	1	1	0	1	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	1	1	0	0	0	1	1	0
5	0	0	1	1	1	0	1	0	0	0	1	0	0	1	0	0	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0
6	0	0	0	1	1	0	0	1	0	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
7	0	0	0	0	1	0	0	0	1	0	1	0	0	1	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	1	0	1	0	1	1

Each test involves 3 shareholders and the secret retrieved by them is to be verified by the AMD decoder. Since holders $\{1, 2, 5\}$ are honest, test 7 is the first test with syndrome 0.

Based on the adaptive Algorithm IV.2, $\triangle T = 7$. The system will only need to run the tests of $\{1, 6, 8, 9\}$ whose participants are $\{1, 2, j\}$ where $j \in H \backslash I = \{3, 4, 6, 7\}$. Thus only tests $\{8, 9\}$ are left to run. The actual number of implemented tests are then $9 < \triangle T + (n-k) \ll \binom{n}{k} = 35$.

In this way the cheaters are identified as: $\{3, 4, 6, 7\}$. And the honest holders $\{1, 2, 5\}$ will be able to retrieve the encoded legal secret $E = 0x3F01$ and therefore $S = 0x3F0$. □

V. Design Analysis of RASSS

A. Error Masking Probability

In Example IV.1 the AMD code works over $GF(2^4)$, per equation [Eq. 8], the error masking probability is $\overline{P_{mask}} = \frac{3}{2^4}$ in the worst case. To increase the security level one can simply have the protocol work over a larger field.

In more of our experiments, the sizes of the encoded secret E are set to $\{8, 16, 32, 48, 64, 80, 96, 128\}$ bits which are the cases for most real-world applications. Therefore, the AMD codes are over $GF(2^b)$ fields where $b \in \{2, 4, 8, 12, 16, 20, 24, 32\}$. A comparison is made between the experimental P_{mask} (under $4 \cdot 2^b$ rounds of RASSS for each b) and the theoretical $\overline{P_{mask}}$.

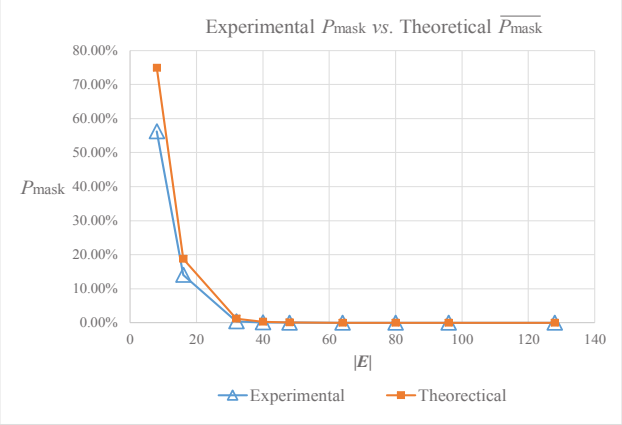

Fig. 2: The experimental P_{mask} matches the theoretical upper bound $\overline{P_{mask}} = \frac{h}{2^b}$. The experimental results are usually better than the upper bound because the left side of equation [Eq. 7] does not always have h solutions in the finite field. Also when $b \geq 32$ the experiments did not miss a single attack.

B. Hardware and Timing Overhead

The hardware cost comparison between RASSS and the classic scheme is made on a Xilinx Vertex 7 XC7VX330T FPGA board under the same parameters as in Section V-A.

The timing comparison is made under severely adverse scenario with $t = n - k$ cheaters for RASSS, and much less cheaters of $t = n/3 - 1$ for the classic scheme. It is implemented by Python on an Intel® Core™ i7-6700 @ 3.4GHz and 8 GB memory machine running Linux OS.

TABLE I: Hardware and Timing Overhead

E (bits)	**Hardware** (Slices)			**Timing** (10^6 clock cycles)		
	Classic	RASSS	Overhead	Classic	RASSS	Overhead
8	521	828	0.59	0.47	3.50	7.38
16	1492	2256	0.51	0.56	5.13	9.17
32	3977	6164	0.55	1.36	14.65	10.75
48	6114	9462	0.55	1.89	22.34	11.81
64	8462	12749	0.51	2.55	27.37	10.75
80	9895	15804	0.59	3.18	32.47	10.21
96	11873	18918	0.59	3.68	40.90	11.12
128	17842	27695	0.55	4.79	50.05	10.44

[I] Overhead = $\frac{\text{RASSS}}{\text{Classic}} - 1$.

[I] With only 60% of the hardware overhead the RASSS protocol drastically improves the cheater tolerance capability. The latency of the classic protocol is 49 logic steps and the latency of RASSS 215 logic steps.

[II] Although the RASSS protocol has a large T as an upper bound, with the adaptive test in Algorithm IV.2 it effectively reduces the actual number of tests on average.

VI. Conclusion

We proposed and implemented a new secure protocol for designing trustworthy distributed systems, called RASSS (Robust Adaptive Secure Secret Sharing). Compared to the classic protocols which can only tolerate up to $t < n/3$ cheaters by RS codes, or detect cheating without cheater tolerance by AMD codes, the RASSS protocol remarkably improves the security level to tolerating $t \leq n - k$ collusive cheaters and retrieving the secret or information as well. When t is beyond this range and even $t = n$, it can still retrieve the secret when provided with additional resources. The adaptivity of the protocol allows an efficient implementation for power sensitive cooperative systems. In future work, we plan to further improve on the practicality of the protocol and significantly reduce the hardware overhead. We also plan to improve the cheater identification and cheating tolerance capability.

References

[1] J. Able *et al.* (2010) Dnssec root zone high level technical architecture. [Online]. Available: http://www.root-dnssec.org/wp-content/uploads/2010/06/draft-icann-dnssec-arch-v1dot4.pdf

[2] Thales, *Microsoft AD CS and OCSP Integration Guide*, 2013.

[3] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22.11, 1979.

[4] R. Cramer *et al.*, "Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors," *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2008.

[5] Z. Wang and M. G. Karpovsky, "Algebraic manipulation detection codes and their applications for design of secure cryptographic devices," *IEEE On-Line Testing Symposium*, 2011.

[6] R. J. McEliece and D. V. Sarwate, "On sharing secrets and reed-solomon codes," *Communications of the ACM*, vol. 24.9, 1981.

[7] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin, "The round complexity of verifiable secret sharing and secure multicast," *33rd annual ACM Symposium on Theory of Computing*, 2001.

[8] M. Fitzi, J. Garay, S. Gollakota, C. P. Rangan, and K. Srinathan, "Round-optimal and efficient verifiable secret sharing," *Theory of Cryptography Conference*, 2006.

[9] E. Berlekamp, *Algebraic coding theory: Revised Edition*. World Scientific, 2015.

[10] S. Gao, "A new algorithm for decoding reed-solomon codes," *Communications, Information and Network Security*, 2003.

[11] H. Krawczyk, "Secret sharing made short," *Annual International Cryptology Conference*, 1993.

[12] J. Liu, S. Mesnager, and L. Chen, "Secret sharing schemes with general access structures," *Intl Conf on Information Security and Cryptology*, 2015.

[13] E. Leducq, "On the third weight of generalized reed-muller codes," *Discrete Mathematics*, 2015.

[14] L. Bu and M. G. Karpovsky, "A design of secure and reliablewireless transmission channel for implantable medical devices," *3rd International Conference on Information Systems Security and Privacy*, 2017.