Revisiting and Extending the AONT-RS scheme: a Robust Computationally Secure Secret Sharing Scheme

Liqun Chen[1], Thalia M. Laing[2], and Keith M. Martin[1]

[1]University of Surrey, Guildford, UK
[2]Information Security Group, Royal Holloway, University of London, Egham, UK
liqun.chen@surrey.ac.uk
{*thalia.laing, keith.martin*}*@rhul.ac.uk*

Abstract

In 2010, Resch and Plank proposed a computationally secure secret sharing scheme, called AONT-RS. We present a generalisation of their scheme and discuss two ways in which information is leaked if used to distribute small ciphertexts. We discuss how to prevent such leakage and provide a proof of computational privacy in the random oracle model. Next, we extend the scheme to be robust and prove the robust AONT-RS achieves computational privacy in the random oracle model and computational recoverability under standard assumptions. Finally, we compare the security, share size and complexity of the AONT-RS scheme with Krawczyk's SSMS scheme.

1 Introduction

A $(t, n)-$threshold secret sharing scheme describes how to distribute data amongst n servers such that t are required to collaborate in order to reconstruct the data. Any fewer than t servers learn nothing about the data. Secret sharing is useful in distributed storage systems where data is stored across multiple servers. A user wishing to access the data must access a threshold number of servers and combine the retrieved shares. This enables greater availability, adds redundancy and offers security without the reliance on cryptographic keys.

In 2010, Resch and Plank proposed a dispersed storage system, called AONT-RS [14], which blends an all-or-nothing transform (AONT) [15] with Reed-Solomon (RS) coding [13]. The result is a computationally secure $(t, n)-$threshold secret sharing scheme. The AONT-RS is a feature in the object storage system sold by Cleversafe, a company recently acquired by IBM [7], who renamed the

product to IBM Cloud Object Storage. In 2016 the system was rated the overall leader in the Gartner Critical Capabilities for Object Storage Report [4].

We present a generalised version of the AONT-RS that enables users the flexibility to choose a block cipher mode of operation and an Information Dispersal Algorithm (IDA). We specify (previously undefined) security properties the building blocks of our generalised AONT-RS must satisfy and discuss information leakage and prevention.

Resch and Plank claim the AONT-RS has integrity because they use a canary, which enables an authorised user to confirm whether or not the correct data has been recovered. However, the submission of an incorrect share will prevent the correct data from being recovered. Although the user knows the recovered data is wrong, they are unable to recover the correct data. To address this, we remove the canary and extend the scheme to be robust by using commitment schemes, as in [16]. This ensures that, even if a bounded number of servers submitted false shares, the original data will be uniquely recovered.

Resch and Plank claim the AONT-RS achieves computational security but no thorough security analysis is provided. We prove both the generalised and robust AONT-RS achieve computational privacy in the random oracle (RO) model, then prove the robust AONT-RS achieves computational recoverability under standard assumptions.

We then compare the generalised AONT-RS with Krawczyk's secret sharing made short scheme (SSMS) [9]. This comparison is applicable to the robust AONT-RS with a robust extension of SSMS by Bellare and Rogaway, called HK2 [16]. In our comparison, we consider the security, the share size and the number of bitwise XORs required to distribute and recover data via both schemes.

1.0.1 Related Work.

Shamir and Blakely independently introduced secret sharing schemes in 1979 [17] [2]. Shamir's scheme is ideal and perfectly secure.

In 1994, Krawczyk published a paper focusing on computationally secure schemes (CSS) in the non-robust setting [9] and proposed his SSMS scheme. Krawczky also proposed goals for a robust CSS scheme, along with a candidate solution. Previously, the CSS goal had been mentioned by Karnin et al. [8]. Prior to Krawczyk's work, robustness had only be studied in the information-theoretic setting in [11] and [18]. Krawczyk's motivation was to achieve shares smaller than were possible in perfectly secure secret sharing schemes [8].

Krawczyk's work was revisited in 2007 by Bellare and Rogaway [16], in which they proposed formal definitions for a CSS and proved Krawczyk's robust scheme to be secure in the RO model. They then proposed a refined version of Krawczyk's scheme (called HK2), which achieves the robust CSS goals under standard assumptions. Since Bellare and Rogaway's work, based on our best knowledge, there have been no new solutions for robust CSS schemes until Resch and Plank's AONT-RS scheme in 2011 [14], which is studied in detail here.

1.0.2 Contributions.

Our contribution can be summarised as follows:

- We present a generalised version of the AONT-RS and highlight the (previously undefined) security properties each element of the scheme must have.
- We discuss and illustrate two examples of information leakage in both the original and our AONT-RS scheme and discuss how to prevent this.
- We prove the AONT-RS achieves computational privacy in the RO model.
- We extend AONT-RS to be robust and provide a proof of computational privacy in the RO model and recoverability under standard assumptions.
- We compare the generalised AONT-RS scheme with Krawczyk's SSMS. Our comparison is applicable to the robust AONT-RS and HK2.

1.0.3 Organisation.

In Section 2 we present notation and definitions. In Section 3 we present a generalised version of the AONT-RS and discuss information leakage. We then prove the generalised AONT-RS achieves computational privacy in the RO model. In Section 4 we extend the AONT-RS to be robust and prove computational privacy in the RO model and recoverability under standard assumptions. In Section 5 we introduce Krawczyk's SSMS and compare it with the generalised AONT-RS. We conclude in Section 6.

2 Preliminaries

In this section we introduce the definitions and notation used throughout.

2.1 Secret Sharing Schemes

Definition 1. *Let $n, t \in \mathbb{N}$ with $2 \leq t \leq n$ and let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n players. A* $(t, n)-$secret sharing scheme Π *consists of two algorithms: Share and Recover. Share is probabilistic and takes as input a* secret s *chosen from a* secret space $\mathcal{S}$ *and outputs an $n-$vector $\boldsymbol{S}$. Player P_i receives the* share $\boldsymbol{S}[i]$. *Recover is deterministic and takes as input shares and outputs some $s' \in \{\mathcal{S} \cup \perp\}$. The secret s should be* recoverable *by any set of at least t players, and* private*, meaning any fewer than t players (called* unauthorised sets*) are unable to recover s.*

A $(t, n)-$secret sharing scheme can either have *perfect* or *computational* security. These security models can be defined by two security games [16] as in Figure 1: one defining privacy and the other recoverability. Note that if an algorithm A is deterministic, we write $x \leftarrow A(\cdot)$. If the algorithm is probabilistic, then $x \xleftarrow{\$} A(\cdot)$ means to choose x according to the distribution induced by A.

GAME *Priv*

- PROCEDURE $Initialise(t, n)$
  $b \xleftarrow{\$} \{0,1\}; \quad j = 1$
- PROCEDURE $Deal(s_0, s_1)$
  If $s_0, s_1 \notin \mathcal{S}$
  Return $\perp$
  Else $\boldsymbol{S} \xleftarrow{\$} Share(s_b)$
- PROCEDURE $Corrupt(i)$
  If $j \leq t-1$
  Return $\boldsymbol{S}[i]; \quad j = j+1$
  Else halt.
- PROCEDURE $Finalise(b')$
  Return $b' = b$

GAME *Rec*

- PROCEDURE $Initialise(t, n)$
  $T \leftarrow \emptyset; \quad j = 1$
- PROCEDURE $Deal(s)$
  If $s \notin \mathcal{S}$
  Return $\perp$
  Else $\boldsymbol{S} \xleftarrow{\$} Share(s)$
- PROCEDURE $Corrupt(i)$
  If $j \leq (n-t)$
  Return $\boldsymbol{S}[i]$
  $j = j+1; \quad T \leftarrow T \cup \{i\}$
  Else halt.
- PROCEDURE $Finalise(\boldsymbol{S}_T)$
  Return
  $s \neq s' \leftarrow Recover(\boldsymbol{S}_T \cup \boldsymbol{S}_{\overline{T}})$

Figure 1: Privacy and recoverability games for a $(t, n)-$secret sharing scheme.

In the privacy game *Priv*, for parameters t and n, the challenger chooses a bit b. The adversary chooses secrets $s_0, s_1 \in \mathcal{S}$ and sends them to the challenger, who inputs s_b to *Share*, which outputs $\boldsymbol{S}$. The adversary then makes up to $t-1$ $Corrupt(i)$ queries for $1 \leq i \leq n$ and receives the share $\boldsymbol{S}[i]$. The adversary then outputs a guess b' for b and wins if $b' = b$.

Let A be an adversary playing the *Priv* game against a secret sharing scheme Π. Call A a *privacy adversary*. Let $\Pr[Priv^A]$ denote the probability A outputs the correct guess $b' = b$ during finalise and define the *advantage* of A as

$$\mathbf{Adv}_{\Pi}^{Priv}(A) = 2 \cdot \Pr[Priv^A] - 1. \tag{1}$$

The recoverability game models an adversary's ability to prevent the recovery of s by deleting or altering shares. The set T denotes the players the adversary corrupts. Let $T = \emptyset$. The adversary chooses and submits a secret $s \in \mathcal{S}$ to the challenger, who inputs it to *Share*. The adversary then makes up to $n - t$ queries of the form $Corrupt(i)$ for $1 \leq i \leq n$ and receives the share $\boldsymbol{S}[i]$ in return. Each i is noted in T. To finalise, the adversary outputs a partially complete $n-$vector $\boldsymbol{S}_T$, consisting of at most t altered (and the rest deleted) shares queried during the corrupt procedure. This vector is completed by the challenger filling the remaining t elements with valid shares noted in the vector $\boldsymbol{S}_{\overline{T}}$. The vector $\boldsymbol{S}_T \cup \boldsymbol{S}_{\overline{T}}$ is then submitted to *Recover*. The adversary wins if $s' \neq s$.

Call adversary A playing the game *Rec* a *recoverability adversary*. Let $\Pr[Rec^A]$ denote the probability s is not correctly recovered. Define the advantage of A as

$$\mathbf{Adv}_{\Pi}^{Rec}(A) = \Pr[Rec^A]. \tag{2}$$

Definition 2. *A* perfectly secure $(t, n)-$threshold scheme (PSS) *is a* $(t, n)-$*threshold scheme in which a privacy adversary and a recoverability adversary restricted to only deleting shares both have an advantage of zero.*

The size of the share $\boldsymbol{S}[i]$ must be at least the size of the secret [1]; if this bound is met the scheme is *ideal*. This bound can be particularly problematic if s is large or the storage available to each player is small. For the application of AONT-RS, we will focus on distributing large amounts of data. Relaxing the security to be computational can achieve smaller shares.

Definition 3. *A* computationally secure $(t, n)-$ scheme (CSS) *is a* $(t, n)-$*threshold secret sharing scheme in which a privacy adversary has a negligible advantage and a recoverability adversary restricted to only deleting (and not corrupting) shares has an advantage of zero.*

CSSs are less secure than PSSs but are able to achieve smaller shares. In general, CSSs are sufficient for most applications [9].

In PSS and CSS schemes, the recoverability adversary is limited to only deleting, and not corrupting, shares. A *robust* scheme ensures the recovery of the secret when the recoverability adversary is allowed to both corrupt and delete a (bounded) number of shares.

Definition 4. *A* robust, computationally secure $(t, n)-$secret sharing scheme *is a* $(t, n)-$*secret sharing scheme in which a privacy adversary and a recoverability adversary both have a negligible advantage at winning their respective games.*

Ramp schemes further relax the security to achieve even smaller shares and are defined information theoretically. Let $\boldsymbol{\mathcal{S}}$ denote the discrete random variable corresponding to the choice of s and let $\boldsymbol{A}$ denote the discrete random variable corresponding to the set of shares given to the players in the set $A \subseteq \mathcal{P}$.

Definition 5. *A* $(t_0, t_1; n)-$ramp scheme *distributes a secret* s *such that any set of at least* t_1 *players can recover* s *and a set of* t_0 *or fewer players reveals no information about the secret. A* $(t_0, t_1; n)-$*ramp scheme is said to be* linear *if, for any set of players* $A \subseteq \mathcal{P}$ *such that* $|A| = r$*, where* $t_0 \leq r \leq t_1$*,*

$$H(\boldsymbol{\mathcal{S}}|\boldsymbol{A}) = \frac{t_1 - r}{t_1 - t_0} H(\boldsymbol{\mathcal{S}}). \tag{3}$$

Note that in a $(t_0, t_1; n)-$linear ramp scheme, for every player after the initial t_0 players have contributed shares, a fixed amount of information is learnt about s. This continues in a linear fashion until t_1 players have contributed and s is learnt completely. Observe that a $(t, n)-$PSS is a $(t - 1, t; n)-$ramp scheme.

2.2 Symmetric Key Encryption

Let $\mathcal{E} = (\mathcal{M}, \mathcal{K}, \mathcal{C}, KenGen,\ Enc, Dec)$ be a symmetric key encryption scheme with message space $\mathcal{M}$, keyspace $\mathcal{K}$, ciphertext space $\mathcal{C}$ and key generation, encryption and decryption algorithms $KeyGen, Enc$ and Dec.

GAME Ind

- PROCEDURE $Initialise$
  $k \xleftarrow{\$} KeyGen\{0,1\}^\lambda$; $b \xleftarrow{\$} \{0,1\}$
- PROCEDURE $Finalise(b')$
  Return $b' = b$
- PROCEDURE $Deal(M_0, M_1)$
  If $M_0 = M_1$, $|M_0| \neq |M_1|$, or $M_0, M_1 \notin \mathcal{M}$
  Return $\perp$
  Else, $C \xleftarrow{\$} Enc_k(M_b)$
  Return C

Figure 2: Game defining indistinguishability in an encryption scheme $\mathcal{E}$

Figure 2 defines the notion of indistinguishability in $\mathcal{E}$ [16]. Call adversary A playing Ind an *indistinguishability adversary.* Let $\Pr[Ind^A]$ denote the probability A outputs the correct guess b'. Define the advantage of A as

$$\mathbf{Adv}^{Ind}_{\mathcal{E}}(A) = 2 \cdot \Pr[Ind^A] - 1 \tag{4}$$

and say $\mathcal{E}$ has indistinguishability if the advantage is negligible.

In Ind, A can repeat the deal procedure multiple times. We can limit A to call the procedure only once; this A is called an *ind-1 adversary.*

2.3 Commitment Schemes

Let $\mathcal{CS}$ be a *commitment scheme* with parameter generation algorithm $ParGen$, commitment algorithm Ct and verification algorithm Vf. Let M denote the message to be committed to, H be a committal and R a decommittal. A commitment scheme should satisfy the hiding and binding properties, defined in two security games in Figure 3 and described in [16]. Intuitively, a commitment scheme allows a sender to commit to a message M and reveal it at a later date.

Call adversary A playing $Hide$ against $\mathcal{CS}$ a *hiding adversary.* Let $\Pr[Hide^A]$ be the probability A correctly guesses $b' = b$. The advantage of A is

$$\mathbf{Adv}^{Hide}_{\mathcal{CS}}(A) = 2 \cdot \Pr[Hide^A] - 1. \tag{5}$$

Say $\mathcal{CS}$ is $\epsilon(\cdot)-$*hiding* if $\mathbf{Adv}^{Hide}_{\mathcal{CS}}(A) \leq \epsilon(q)$ for any adversary that makes at most q queries during the deal procedure.

Call adversary A playing $Bind$ a *binding adversary.* The advantage of A is

$$\mathbf{Adv}^{bind}_{\mathcal{CS}}(A) = \Pr[Bind^A]. \tag{6}$$

2.4 Error Correcting Codes

An error correcting code (ECC) E is a method of encoding data with some redundant information to ensure the original data can be recovered, even if a number of errors occur during either data transmission or storage [10].

GAME *Hide*

- PROCEDURE *Initialise*
  $\pi \xleftarrow{\$} ParGen$; $b \xleftarrow{\$} \{0,1\}$
- PROCEDURE $Deal(M_0, M_1)$
  If $M_0, M_1 \notin \mathcal{M}$
  Return $\perp$
  Else $(H, R) \xleftarrow{\$} Ct(\pi, M_b)$
  Return H
- PROCEDURE $Finalise(b')$
  Return $b' = b$

GAME *Bind*

- PROCEDURE *Initialise*
  $\pi \xleftarrow{\$} ParGen$
- PROCEDURE $Commit(M_0)$
  If $M_0 \notin \mathcal{M}$
  Return $\perp$.
  Else $(H, R_0) \xleftarrow{\$} Ct(\pi, M_0)$
  Return (H, R_0)
- PROCEDURE $Finalise(M_1, R_1)$
  If $M_1 \notin \mathcal{M}$, return $\perp$.
  Return $M_0 \neq M_1$ and
  $Vf(H, M_0, R_0) =$
  $Vf(H, M_1, R_1) = 1$

Figure 3: Games defining the hiding and binding security properties of $\mathcal{CS}$.

A code E of *length* n over a finite alphabet F is a subset of F^n. Elements of E are called *codewords*. The *size* of E is $|E| = m$. The *minimum distance* d is the minimum Hamming distance between any two distinct codewords.

Let E be a *linear* code, meaning that for all $u, w \in E$, we have $u + w \in E$, where addition is modulo q with $|F| = q$. If $u_1, \ldots, u_t$ is a basis for E, then E has *dimension* t. There are q^t possible codewords and we call E a $[n, t, d]-$*code*.

One important ECC is a *maximum distance separable* (MDS) code [10], which is a linear code that meets the Singleton bound: $d = n - t + 1$. For any MDS code, recovery of a codeword is possible from any t of the n symbols. Denote such a code as $(t, n)-$ECC. A Reed Solomon (RS) code [13] is an MDS code. A code where message string appears in the codeword is called *systematic*.

Let $\boldsymbol{U} \xleftarrow{\$} Share^{ECC}(u)$ denote the distribution of a word u via a $(t, n)-$ECC, resulting in a codeword represented by an $n-$vector $\boldsymbol{U}$. The word u is recoverable from any t elements of $\boldsymbol{U}$ via the deterministic algorithm $Recover^{ECC}$.

2.5 Information Dispersal Algorithms

Information dispersal was first introduced by Rabin [12].

Definition 6. *Let $t, n \in \mathbb{N}, t \leq n$. A $(t, n)-$*information dispersal algorithm *(IDA) with message space $\mathcal{M}$ consists of two algorithms $Share^{IDA}$ and $Recover^{IDA}$. $Share^{IDA}$ takes as input a message $M \in \mathcal{M}$ and outputs an $n-$vector $\boldsymbol{S}$. $Recover^{IDA}$ takes as input elements of the vector $\boldsymbol{S}$. If at least t elements are submitted correctly to $Recover^{IDA}$, the algorithm outputs the message M.*

A $(t, n)-$IDA shares M between n players such that any t players can recover M. This is equivalent to the recoverability property of a $(t, n)-$secret sharing scheme. A $(t, n)-$secret sharing scheme satisfies the conditions of an IDA but

additionally guarantees privacy, which IDAs do not. IDAs are able to achieve smaller share sizes by taking advantage of the lack of privacy.

2.5.1 Resch and Plank's IDA.

In the AONT-RS scheme, a systematic IDA, which is a variant of an RS code [14], is specified. We refer to this as the *systematic RS IDA* with algorithms $Share^{RS-IDA}$ and $Recover^{RS-IDA}$.

Let $F = GF(2^\omega)$ be a Galois field of characteristic 2. $Share^{RS-IDA}$ is a probabilistic algorithm that takes as input M and parses M into t words, treating this as a $t-$vector, $\boldsymbol{M} \in F^t$. This vector is multiplied on the left by a public $n \times t$ binary matrix G, where multiplication of elements $b \in \{0,1\}$ and $d \in F$ is defined as: $\{0,1\} \times F \to F$, where $0 \times d = 0 \in F$ and $1 \times d = d \in F$. The matrix G is constructed such that the first t rows form the $t \times t$ identity matrix and any t of the n rows are linearly independent; the last $n-t$ rows can be generated in any manner as long as this condition is satisfied. The resulting n vector is the codeword vector $G \cdot \boldsymbol{M} = \boldsymbol{V} \in F^n$. Each player receives the share $\boldsymbol{V}[i]$.

In order to recover M, t shares are submitted to $Recover^{RS-IDA}$ and a $t-$vector $\boldsymbol{V'}$ is created from these shares. A $t \times t$ matrix G' is formed, consisting of the t rows of G corresponding to the shares pooled. This matrix is inverted and multiplied by $\boldsymbol{V'}$ to return $(G')^{-1} \cdot \boldsymbol{V'} = \boldsymbol{M}$, from which M can be constructed.

It is known that an RS code, which is a $[n, t, n-t+1]-$code, is equivalent to a $(0, t; n)-$linear ramp scheme [5]. Thus the systematic RS IDA used by Resch and Plank is equivalent to a $(0, t; n)-$linear ramp scheme, as in Definition 5.

3 The AONT-RS

In this section, we consider the Resch and Plank's AONT-RS scheme [14]. We present a generalised version, then discuss information leakage. Finally, we prove the AONT-RS scheme achieves computational privacy in the RO model.

3.1 Generalising the AONT-RS

Resch and Plank propose a CSS in [14], which they call AONT-RS. It combines an All or Nothing Transform (AONT) with an RS code. An AONT is an encryption mode that allows the data to be learnt only if all of it is known [3].

Their scheme assumes the existence of a symmetric key encryption scheme $\mathcal{E}$ operating on blocks of plaintext in CBC mode, a cryptographic hash function H and the systematic RS IDA, as in Section 2.5.1. They assume the digest of the hash function is of equal length to the key k use in $\mathcal{E}$. They do not define what security properties $\mathcal{E}$ must have. They also use a canary, which is a known, fixed value concatenated with the plaintext. When a message is recovered, the user can compare the recovered value with the known canary to verify correctness.

We observe that $\mathcal{E}$ requires ind-1 security and must be probabilistic, but need not operate in CBC mode. We allow flexibility of the IDA, as long as it is equivalent to a $(0, t; n)-$linear ramp scheme (which the systematic RS IDA is). We remove the concept of the canary from the definition, noting that if an incorrect message were recovered, a canary would not help recover the correct M. Preventing M from being incorrectly recovered is discussed in Section 4.

Let Π denote our generalised AONT-RS scheme, with algorithms defined in Figure 4. From now on, AONT-RS will refer to algorithms in Figure 4.

PROCEDURE $Share^{AONT}(M)$

1. $k \xleftarrow{\$} KeyGen(\{0,1\}^{\lambda})$
   $C \xleftarrow{\$} Enc_k(M)$
2. $h = H(C); \quad c_d = h \oplus k$
3. $\boldsymbol{V} \leftarrow Share^{IDA}(C||c_d)$
4. Return $\boldsymbol{V}$

PROCEDURE $Recover^{AONT}(M)$

1. $\boldsymbol{V} \leftarrow$ $Recover^{IDA}(\boldsymbol{V}[0], \ldots, \boldsymbol{V}[n-1])$
2. $C||c_d \leftarrow \boldsymbol{V}$
3. $h = H(C); \quad k = h \oplus c_d$
4. $M \leftarrow Dec_k(C)$
5. Return M

Figure 4: The dispersal and recovery algorithms defining the AONT-RS scheme.

On input $M \in \mathcal{M}$, $Share^{AONT}$ generates a key k of length λ, encrypts M under k, then computes the hash of the ciphertext $h = H(C)$. The digest h is then XORed with k to give a value c_d, which we call the *difference value*. The difference value and C are concatenated and dispersed via an IDA.

To recover M, at least t players must pool their shares into a vector $\boldsymbol{V}$. Using the algorithm $Recover^{IDA}$, $C||c_d$ is recovered. The digest $h = H(C)$ is calculated and XORed with c_d to recover k, which is used to decrypt C and return M.

If $\mathcal{E}$ were not probabilistic, an adversary may recognise shares of known ciphertexts and be able to predict C which, if c_d is known, could leak information about k. The scheme also requires ind-1 security; general indistinguishability is not required as each time a new M is shared, a new encryption key k is generated. So each key is only used to encrypt one message.

3.2 Information Leakage

Resch and Plank claim their system is secure because $t-1$ players are unable to recover all of $\boldsymbol{V}$, due to the security of the IDA. Without all of $\boldsymbol{V}$, players are unable to learn both C (required to compute $h = H(C)$) and c_d and so learn nothing about k and M. Learning either C or c_d in isolation does not help the adversary. In order to recover M, knowledge of both C and c_d are needed.

It is necessary that an unauthorised set learn at most: some or all of c_d and some (but not all) of C, or none of c_d and all of C. We show that, when C is a short ciphertext (in relation to the security parameter λ and the threshold

value t), the adversary may be able to learn enough to leak information about k.

3.2.1 Learning C completely and c_d partially.

Consider the following example. Let $C, k \in \{0,1\}^{128}$. Let there be $n = 5$ players $P_1, \ldots, P_5$ and let $t = 4$. The string $C||c_d$ would be parsed into four words to make the $t-$vector $\boldsymbol{M}$, where each fragment is 64 bits. Let c_0 and c_1 be the two elements that comprise C and let $c_{d,0}$ and $c_{d,1}$ be the two halves of c_d, each 64 bits. The vector $\boldsymbol{M}$ is then multiplied on the left by the generator matrix $G \in \{0,1\}^{(5\times 4)}$, which gives

$$G \cdot \boldsymbol{M} == \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ G_{4,0} & G_{4,1} & G_{4,2} & G_{4,3} \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ c_{d,0} \\ c_{d,1} \end{pmatrix} = \begin{pmatrix} C_0 \\ C_1 \\ c_{d,0} \\ c_{d,1} \\ x \end{pmatrix},$$

where $G_{i,0} \xleftarrow{\$} \{0,1\}$, for $i = \{0, \ldots, 3\}$ are chosen such that any 4 rows of G are linearly independent and $x = G_{4,0} \cdot C_0 + G_{4,1} \cdot C_1 + G_{4,2} \cdot c_{d,0} + G_{4,3} \cdot c_{d,1}$.

Players P_1, P_2 and P_3 are an unauthorised set, yet they could learn all of C and $c_{d,0}$. They could then compute $H(C) = h$ and XOR the first half of h with $c_{d,0}$ to recover the first half of k. This reduces the security from 128 to 64 bits.

This attack can be prevented if c_d is contained entirely in one share. So if $c_d \in \{0,1\}^{\lambda}$, then C should be such that $C \in \{0,1\}^{\omega}$, where $\omega \geq (t-1)\lambda$.

3.2.2 Learning C partially and c_d completely.

An alternative version of this attack utilises the fact that the hash function H is deterministic.

Consider the following example. Assume an attacker knows all of c_d and all but one bit of C. They can construct two possibilities for C (C_0 when the unknown bit is 0, and C_1 when it is 1) and compute the corresponding hashes, $h_0 = H(C_0)$ and $h_1 = H(C_1)$. They can then compute two key candidates $k_0 = c_d \oplus h_0$ and $k_1 = c_d \oplus h_1$ and decrypt the ciphertexts C_0 and C_1 with the corresponding candidate keys to reveal two plaintext messages M_0 and M_1. From these, the adversary can guess which plaintext message is likely to be the true message and has thus learnt k. In general, if the adversary knows c_d and all but j bits of C, if $j < \lambda$ this attack is quicker than brute force. We must ensure an adversary is unable to learn at least λ bits of C if c_d is known. This is true if each $\boldsymbol{M}[i] \in \{0,1\}^{\lambda}$, meaning that $C \in \{0,1\}^{\omega}, \omega \geq (t-1)\lambda$.

So both attacks can be prevented if $k \in \{0,1\}^{\lambda}$ and $C \in \{0,1\}^{\omega}$, where $\omega \geq (t-1)\lambda$. If C is too small, C should be padded with some random string. This condition on the size of C is a necessary, but not sufficient, condition for the AONT-RS scheme to be secure. To guarantee the security, we must make additional assumptions on H, $\mathcal{CS}$ and $\mathcal{E}$, as discussed next.

PROCEDURE *Initialise* G_0, G_1, G_2
$k \xleftarrow{\$} \{0,1\}^\lambda$; $b \xleftarrow{\$} \{0,1\}$; $k' = k$

PROCEDURE *Deal*(x_0, x_1)
G_0, G_1, G_4, G_5
$C \leftarrow Enc_k(x_b)$; $H(C) = h$;
$h \oplus k' = c_d$
$\boldsymbol{V} \leftarrow Share^{IDA}(C||c_d)$
For $i \leftarrow 1$ to n do
$(\boldsymbol{H}[i], \boldsymbol{R}[i]) \xleftarrow{\$} Ct(\boldsymbol{V}[i])$
$\boldsymbol{S}_i \xleftarrow{\$} Share^{ECC}(\boldsymbol{H}[i])$

PROCEDURE *Corrupt*(i) G_0, G_5
$\boldsymbol{X}[i] \leftarrow \boldsymbol{R}[i]\boldsymbol{V}[i]\boldsymbol{S}_1[i]...\boldsymbol{S}_n[i]$
Return $\boldsymbol{X}[i]$

PROCEDURE *Finalise*(b') $G_0 - G_5$
Return $(b' = b)$

PROCEDURE *Initialise* G_3, G_4, G_5
$k, k' \xleftarrow{\$} \{0,1\}^\lambda$; $b \xleftarrow{\$} \{0,1\}$

PROCEDURE *Deal*(x_0, x_1) G_2, G_3
$C \leftarrow Enc_k(x_b)$; $\boldsymbol{C} \leftarrow Share^{IDA}(C||0)$
For $i \leftarrow 1$ to n do
$(\boldsymbol{H}[i], \boldsymbol{R}[i]) \xleftarrow{\$} Ct(\boldsymbol{C}[i])$
$\boldsymbol{S}[i] \leftarrow Share^{ECC}(\boldsymbol{H}[i])$
$H(C) = h$; $h \oplus k' = c_d$
$\boldsymbol{V} \leftarrow Share^{IDA}(C||c_d)$

PROCEDURE *Corrupt*(i) G_1, G_2, G_3, G_4
$\boldsymbol{R}[i] \xleftarrow{\$} DCt(\boldsymbol{H}[i], \boldsymbol{V}[i])$
$\boldsymbol{X}[i] \leftarrow \boldsymbol{R}[i]\boldsymbol{V}[i]\boldsymbol{S}_1[i]...\boldsymbol{S}_n[i]$
Return $\boldsymbol{X}[i]$

Figure 5: Games G_0 and G_5 are used to prove Theorem 1, the privacy of AONT-RS. All games $G_0 - G_5$ are used to prove Theorem 2, the privacy of the RAONT-RS.

3.3 Proving the Privacy of AONT-RS

We now prove the AONT-RS achieves computational privacy in the RO model.

Theorem 1 (Privacy of the non-robust AONT-RS)**.** *Let A be a privacy adversary against the AONT-RS scheme Π (as in Figure 4) and let the internal hash function H be indistinguishable from a RO. Let the ciphertext be $C \in \{0,1\}^\omega$, where $\omega \geq (t-1)\lambda$. Then there is an ind-1 adversary B attacking the indistinguishability of $\mathcal{E}$ such that*

$$\mathbf{Adv}^{Priv}_{\Pi}(A) \leq \mathbf{Adv}^{Ind}_{\mathcal{E}}(B), \tag{7}$$

where B makes only one query during the deal procedure of Game Ind and the running time of B is that of A plus one execution of $Share^{AONT}$.

Proof. The proof relies on games G_0 and G_5, as in Figure 5. The figure shows multiple procedures, indicating next to each in which games it is included. For example G_0 is defined by the procedures on the left hand side of the figure. The advantage of the AONT-RS privacy adversary A can be defined as

$$\mathbf{Adv}^{Priv}_{\Pi}(A) = 2 \cdot \Pr[G^A_0] - 1. \tag{8}$$

Game G_5 differs from G_0 only because the key k used to encrypt M is different to the value k' used to compute $c_d = h \oplus k'$. We claim that $\Pr[G^A_0] =$

$\Pr[G_5^A]$, as the hash function H is indistinguishable from a RO. Due to the IDA used and the restriction that $C = \{0,1\}^\omega$, where $\omega \geq \lambda(t-1)$, the adversary is always missing either at least λ bits of C or all of c_d. Thus the adversary can learn either h or c_d. If A learns h, $h = c'_d \oplus k'$. If A learns c_d, then $c_d = h' \oplus k'$, for some $c'_d \neq c_d$ and $h' \neq h$. Thus $\Pr[G_0^A] = \Pr[G_5^A]$ holds true.

We construct an adversary B attacking the privacy of $\mathcal{E}$ such that

$$2 \cdot \Pr[G_5^A] - 1 \leq \mathbf{Adv}_{\mathcal{E}}^{Ind}(B). \tag{9}$$

Adversary B picks $k' \xleftarrow{\$} \{0,1\}^\lambda$ and runs A. A submits $Deal(x_0, x_1)$ to B, who queries x_0, x_1 to its challenger and receives $C \xleftarrow{\$} Enc_k(x_b)$, where k is the key generated by the challenger. Now B executes the rest of the $Deal$ procedure of G_5 using k'; so B computes $H(C) = h$, $h \oplus k' = c_d$ and $\boldsymbol{V} \leftarrow Share^{IDA}(C||c_d)$. When A submits $Corrupt(i)$, B responds with $\boldsymbol{V}[i]$. When A outputs a bit b', B passes this onto their challenger. The advantage of B is $2 \cdot \Pr[b' = b] - 1$.

By combining (8), (9) and $\Pr[G_0^A] = \Pr[G_5^A]$, we see that, as required,

$$\mathbf{Adv}_{\Pi}^{Priv}(A) \leq \mathbf{Adv}_{\mathcal{E}}^{Ind}(B).$$

□

4 Extending AONT-RS to be Robust

In [16], Bellare and Rogaway extend Krawczyk's SSMS [9] to be robust by using commitment schemes. Their technique can be applied to the AONT-RS to make it robust. We will call the resulting scheme the RAONT-RS.

Let $\mathcal{E}$ be an ind-1 secure encryption scheme. Assume the existence of a $(t,n)-$ECC, an IDA equivalent to a $(0,t;n)-$linear ramp scheme, an $\epsilon(\cdot)-$hiding commitment scheme $\mathcal{CS}$ and a hash function H that is indistinguishable from a RO. Let Π_R denote the RAONT-RS scheme, with algorithms as in Figure 6. Let the ciphertext be $C \in \{0,1\}^\omega$, where $\omega \geq (t-1)\lambda$.

Intuitively, the scheme is the same as the AONT-RS. However, in addition to being given the share $\boldsymbol{V}[i]$, each player is given a decommittal $\boldsymbol{R}[i]$ computed on $\boldsymbol{V}[i]$ and fragments of committals $\boldsymbol{H}[i]$ computed on all shares $\boldsymbol{V}[i]$ distributed via a $(t,n)-$ECC. Let the n-vector $\boldsymbol{S}_i$ be the output after the committal $\boldsymbol{H}[i]$ is dispersed via a $(t,n)-$ECC . Let $\boldsymbol{S}_i[j]$ be the j^{th} element of $\boldsymbol{S}_i$. These values are used to verify each player's share. Let $\Diamond$ denote an empty share.

Unlike a canary, commitment schemes allow recovery of M even if false shares are submitted. Furthermore, the commitment scheme highlights which servers are corrupted and thus take any necessary action. However, it is noted that commitment schemes requires more computation than the use of a canary. In practise, both techniques could be combined; the canary could first be verified and, only if the canary is incorrect, will the shares be individually verified.

PROCEDURE $Share^{\mathrm{RAONT}}(M)$

1. $k \xleftarrow{\$} \{0,1\}^{\lambda}$; $\quad C \xleftarrow{\$} Enc_k(M)$
2. $h = H(C)$; $\quad c_d = h \oplus k$
3. $\boldsymbol{V} \leftarrow Share^{IDA}(C||c_d)$
4. For $i \leftarrow 1$ to n do
   $(\boldsymbol{H}[i], \boldsymbol{R}[i]) \xleftarrow{\$} Ct(\boldsymbol{V}[i])$
   $\boldsymbol{S}_i \xleftarrow{\$} Share^{ECC}(\boldsymbol{H}[i])$
5. For $i \leftarrow 1$ to n do
   $\boldsymbol{X}[i] \leftarrow$
   $\boldsymbol{R}[i]\boldsymbol{V}[i]\boldsymbol{S}_1[i] \ldots \boldsymbol{S}_n[i]$
6. Return $\boldsymbol{X}$

PROCEDURE $Recover^{\mathrm{RAONT}}(\boldsymbol{V})$

1. For $i \leftarrow 0$ to $n-1$ do
   $\boldsymbol{R}[i]\boldsymbol{V}[i]\boldsymbol{S}_1[i] \ldots \boldsymbol{S}_n[i] \leftarrow \boldsymbol{X}[i]$
2. For $i \leftarrow 0$ to $n-1$ do
   $\boldsymbol{H}[i] \leftarrow Recover^{ECC}(\boldsymbol{S}_i, j)$
3. For $i \leftarrow 0$ to $n-1$ do
   If $\boldsymbol{X}[i] \neq \Diamond$ and
   $Vf(\boldsymbol{H}[i], \boldsymbol{V}[i], \boldsymbol{R}[i]) = 0$
   then $\boldsymbol{V}[i] \leftarrow \Diamond$
4. $C||c_d \leftarrow Recover^{IDA}(\boldsymbol{V})$
5. $h = H(C)$; $\quad k = h \oplus c_d$
6. $M \leftarrow Dec_k(C)$
7. Return M

Figure 6: Algorithms defining robust AONT-RS (RAONT-RS).

4.1 Proof of Privacy

The RAONT-RS scheme Π_R can be proven to achieve computational privacy by adapting the proof of privacy for the HK2 scheme by Bellare and Rogaway [16].

Theorem 2 (Privacy of RAONT-RS)**.** *Let A be a privacy adversary against RAONT-RS Π_R and let H be indistinguishable from a RO. Let the ciphertext be $C \in \{0,1\}^{\omega}$, where $\omega \geq (t-1)\lambda$. Assume Ct is $\epsilon(\cdot)-$hiding (as in Section 2.3), then there is an ind-1 adversary B attacking the indistinguishability of $\mathcal{E}$ such that*

$$\mathbf{Adv}^{Priv}_{\Pi_R}(A) \leq \mathbf{Adv}^{Ind}_{\mathcal{E}}(B) \cdot 4\epsilon(n), \tag{10}$$

where B makes only one query during the deal procedure of Game Ind and the running time of B is that of A plus one execution of $Share^{RAONT}$.

Proof. The proof relies on games $G_0 - G_5$, as in Figure 5. The procedure *Corrupt* of games $G_1 - G_4$ refers to a probabilistic algorithm DCt that works as follows. On input message M and committal H, it lets $\Omega(M, H)$ denote the set of all coins ω such that Ct, on input M and coins ω, returns a pair whose first component is H. If $\Omega(M, H) = \emptyset$, then DCt returns $\perp$. Else it picks ω at random from $\Omega(M, H)$, runs Ct on input M and coins ω to get a pair (H, R) and returns R. This algorithm is not necessarily efficiently implementable.

The advantage of the RAONT-RS privacy adversary A can be defined as

$$\mathbf{Adv}^{Priv}_{\Pi_R}(A) = 2 \cdot \Pr[G^A_0] - 1. \tag{11}$$

Game G_1 differs from G_0 only in the *Corrupt* procedure, which resamples $\boldsymbol{R}[i]$ using DCt. Clearly,

$$\Pr[G^A_0] = \Pr[G^A_1] = \Pr[G^A_2] + (\Pr[G^A_1] - \Pr[G^A_2]). \tag{12}$$

We construct an adversary D_1 attacking the hiding property of $\mathcal{CS}$ such that

$$\Pr[G^A_1] - \Pr[G^A_2] = \mathbf{Adv}^{Hide}_{\mathcal{CS}}(D_1). \tag{13}$$

Adversary D_1 acts as the challenger to A and wishes to use A's advantage to gain an advantage against the hiding property of $\mathcal{CS}$. Adversary D_1 picks $b \xleftarrow{\$} \{0,1\}$ and runs A. When A submits x_0, x_1 to D_1, D_1 generates $k \xleftarrow{\$} \{0,1\}^\lambda$ and calculates $C \xleftarrow{\$} Enc_k(x_b)$. D_1 then computes $H(C) = h$ and $h \oplus k = c_d$, then calculates both $\boldsymbol{V} \leftarrow Share^{IDA}(C||c_d)$ and $\boldsymbol{C} \leftarrow Share^{IDA}(C||0)$. For i, $1 \leq i \leq n$, D_1 queries $\boldsymbol{C}[i], \boldsymbol{V}[i]$ (for $\boldsymbol{V}[i] \neq \boldsymbol{C}[i]$) to its challenger. Let $\boldsymbol{H}[i]$ denote the commitment value returned. Let $\boldsymbol{S}_i \leftarrow Share^{ECC}(\boldsymbol{H}[i])$. When A makes a $Corrupt(i)$ query to D_1, D_1 computes its reply according to the case of the *Corrupt* procedure of games G_1, G_2; that is, D_1 generates a decommittal value $\boldsymbol{R}[i]$ for $\boldsymbol{V}[i]$ and the given $\boldsymbol{H}[i]$ and passes $\boldsymbol{X}[i] \leftarrow \boldsymbol{R}[i]\boldsymbol{V}[i]\boldsymbol{S}_1[i] \ldots \boldsymbol{S}_n[i]$ to A. When A halts the corruption procedure and finalises with output b', if $b' = b$, adversary D_1 passes 1 to its challenger, guessing the commitment value $\boldsymbol{H}[i]$ was computed on $\boldsymbol{V}[i]$, rather than $\boldsymbol{C}[i]$. Otherwise, D_1 submits 0.

Next, we have that

$$\Pr[G^A_2] = \Pr[G^A_3] + \left(\Pr[G^A_2] - \Pr[G^A_3]\right), \tag{14}$$

where G_3 differs from G_2 only in the initialise procedure which XORs the digest h not with the encryption key k, but with a string k'. We claim that $\Pr[G^A_2] = \Pr[G^A_3]$. because the hash function is indistinguishable from a RO. After A has corrupted at most t shares, they learn at most either

- no information about c_d and all of C, and so can learn $h = H(C)$. In which case $h = k \oplus c_d = k' \oplus c'_d$, where $c'_d \neq c_d$ is some unknown string. Or
- all of c_d, but missing at least λ bits of C. Then $c_d = k' \oplus h'$ where $h' \neq h$ is unknown to A.

In either case, the adversary learns either h or c_d and no information about k. Thus the known value is the XOR of two unknown strings: changing one of these strings does not affect the chances of A winning, thus $\Pr[G^A_2] = \Pr[G^A_3]$.

Next, we have

$$\Pr[G^A_3] = \Pr[G^A_4] + \left(\Pr[G^A_3] - \Pr[G^A_4]\right). \tag{15}$$

Construct adversary D_2, also attacking the hiding property of $\mathcal{CS}$, such that

$$\Pr[G^A_3] - \Pr[G^A_4] = \mathbf{Adv}^{Hide}_{\mathcal{CS}}(D_2). \tag{16}$$

PROCEDURE $Deal(x)$
$\ell \xleftarrow{\$} [1,n]$; $k \xleftarrow{\$} \{0,1\}^\lambda$; $C \xleftarrow{\$} Enc_k(x)$
$H(c) = h$; $h \oplus k = c_d$; $\boldsymbol{V} \leftarrow Share^{IDA}(C||c_d)$
For $i \leftarrow 1$ to n do
  If $i = \ell$, then $(\boldsymbol{H}[\ell], \boldsymbol{R}[\ell]) \xleftarrow{\$} Commit(\boldsymbol{V}[i])$
  Else $(\boldsymbol{H}[i], \boldsymbol{R}[i]) \xleftarrow{\$} Ct(\boldsymbol{V}[i])$
  $\boldsymbol{S}_i \xleftarrow{\$} Share^{ECC}(\boldsymbol{H}[i])$
For $i \leftarrow 1$ to n do
  $\boldsymbol{X}[i] \leftarrow \boldsymbol{R}[i]\boldsymbol{V}[i]\boldsymbol{S}_i[i] \dots \boldsymbol{S}_n[i]$

PROCEDURE $Corrupt(i)$
Return $\boldsymbol{X}[i]$

PROCEDURE $Finalise(\boldsymbol{x}', j)$
For $i \leftarrow 1$ to n do
  $\boldsymbol{R}'[i]\boldsymbol{V}'[i]\boldsymbol{S}'_1[i] \dots \boldsymbol{S}'_n[i] \leftarrow \boldsymbol{X}'[i]$
Return $(\boldsymbol{V}'[\ell], \boldsymbol{R}'[\ell])$

Figure 7: Procedures used by adversary B to respond to A for Theorem 3.

The construction of D_2 is similar to D_1, but D_2 generates $k, k' \xleftarrow{\$} \{0,1\}^\lambda$, encrypts x_b under k as before and now calculates $c_d = h \oplus k'$.

Game G_5 and G_4 differ only during *Corrupt*. Clearly $\Pr[G_4^A] = \Pr[G_5^A]$.

Let B be an ind-1 adversary attacking $\mathcal{E}$, as in the proof of Theorem 1. The advantage of B is as in (9).

Now, let D be the hiding-adversary that flips a fair coin and, if it lands head, runs D_1, otherwise D_2. Clearly,

$$\mathbf{Adv}^{Hide}_{\mathcal{CS}}(D) = \frac{1}{2}\left(\mathbf{Adv}^{Hide}_{\mathcal{CS}}(D_1) + \mathbf{Adv}^{Hide}_{\mathcal{CS}}(D_2)\right). \tag{17}$$

Since Ct is assumed to be $\epsilon(\cdot)-$hiding and D makes at most n queries, we have that $\mathbf{Adv}^{Hide}_{\mathcal{CS}}(D) \leq \epsilon(n)$. Combining this and (13), (16), (17) gives us

$$\left(\Pr[G_1^A] - \Pr[G_2^A]\right) + \left(\Pr[G_3^A] - \Pr[G_4^A]\right) \leq 2\epsilon(n).$$

By using $\Pr[G_2^A] = \Pr[G_3^A]$ and $\Pr[G_4^A] = \Pr[G_5^A]$ and substituting in the advantages of adversaries A and B, we can simplify and rearrange to give

$$\mathbf{Adv}^{Priv}_{\Pi_R}(A) \leq \mathbf{Adv}^{Ind}_{\mathcal{E}}(B) \cdot 4\epsilon(n),$$

thus completing the proof.

□

4.2 Proof of Robustness

The RAONT-RS scheme can be proven to be robust by adapting the proof by Bellare and Rogaway [16].

Theorem 3 (Robustness of RAONT-RS)**.** *Let A be a recoverability adversary against the RAONT-RS scheme Π_R. Then there is an adversary B attacking the binding property of the commitment scheme $\mathcal{CS}$ such that*

$$\mathbf{Adv}^{Rec}_{\Pi_R}(A) \leq n \cdot \mathbf{Adv}^{Bind}_{\mathcal{CS}}(B), \tag{18}$$

where the running time of B is that of A plus overhead consisting of an execution of the $Share^{RAONT}$ and $Recover^{RAONT}$ algorithms of Π_R.

Proof. Let A be a recoverability adversary against Π_R. During *Deal*, A submits x to B. Let $k, C, h, c_d, \boldsymbol{V}, \boldsymbol{H}, \boldsymbol{S_1}, \ldots, \boldsymbol{S}_n, \boldsymbol{X} \xleftarrow{\$}$ denote the quantities chosen by $Share^{RAONT}(x)$. Let A corrupt at most $t-1$ shares. Let $(\boldsymbol{X}_T)$ denote the output of A. Let $k', C', h', c'_d, \boldsymbol{V}', \boldsymbol{H}', \boldsymbol{S'_1}, \ldots, \boldsymbol{S}'_n, \boldsymbol{X}'$ denote the quantities recovered from $Recover^{RAONT}$ with input $\boldsymbol{X}'_T \cup \boldsymbol{X}'_{\overline{T}}$. Consider the following events:

E_1: $\exists \ell \in [n]$ such that $\boldsymbol{H}[\ell] \neq \boldsymbol{H}'[\ell]$
E_2: $\exists \ell \in T$ such that $\boldsymbol{V}[i] \in \{\Diamond, \boldsymbol{V}[i]\}$
E_3: $c_d \neq c'_d$
E_4: $C \neq C'$

If $C' = C$ and $c'_d = c_d$, then the recovered secret x' equals x. This is because $h' = H(C') = H(C) = h$ and so $c'_d \oplus h' = c_d \oplus h = k$. Therefore

$$\begin{aligned}
\mathbf{Adv}^{Rec}_{\Pi_R}(A) &\leq \Pr[E_3 \cup E_4] && (19)\\
&\leq \Pr[E_1 \cup E_2 \cup E_3 \cup E_4] && (20)\\
&= \Pr[E_1] + \Pr[\overline{E_1} \cap E_2] + \Pr[\overline{E_2} \cap E_3] + \Pr[\overline{E_2} \cap E_4]. && (21)
\end{aligned}$$

We bound each addend in turn. Let $E_{1,\ell}$ be the event that $\boldsymbol{H}'[\ell] = \boldsymbol{H}[\ell]$. Let T be the set of indexes of the shares corrupted by A. If $i \notin T$, then the submission of $\boldsymbol{X}'[i]$ and the other uncorrupted shares returns $\boldsymbol{X}[i]$. Hence $\boldsymbol{S}'_\ell[i] = \boldsymbol{S}_\ell[i]$. Note that $\boldsymbol{S}_\ell$ is an output of $Share^{ECC}(\boldsymbol{H}[\ell])$. Lemma 10 in [16] discusses perfect recoverability and, when applied to ECCs, $Recover^{ECC}(\boldsymbol{S}_\ell) = \boldsymbol{H}[\ell]$, meaning that $\boldsymbol{H}'[\ell] = \boldsymbol{H}[\ell]$. So $\Pr[E_{1,\ell}] = 0$. By the union bound

$$\Pr[E_1] \leq \sum_{t=1}^{n} \Pr[E_{1,\ell}] = 0. \tag{22}$$

Now we construct adversary B such that

$$\Pr[\overline{E_1} \cup E_2] \leq n \cdot \mathbf{Adv}^{Bind}_{\mathcal{CS}}(B). \tag{23}$$

Adversary B runs A, responding to its *Deal* and *Corrupt* calls via the procedures in Figure 7, where Ct is the committal algorithm of $\mathcal{CS}$ run by B and *Commit* is a procedure of the *Bind* game that B plays with its challenger. When A halts with output $(\boldsymbol{X})$, B runs the finalise procedure.

Next, we claim both $\Pr[\overline{E_2} \cap E_3] = 0$ and $\Pr[\overline{E_2} \cap E_4] = 0$. As, if $\boldsymbol{V}'[i] = \boldsymbol{V}[i]$ for all i, then $C = C'$ and $c'_d = c_d$. So now

$$\begin{aligned}
\mathbf{Adv}^{Rec}_{\Pi_R}(A) &= \Pr[E_1] + \Pr[\overline{E_1} \cap E_2] + \Pr[\overline{E_2} \cap E_3] + \Pr[\overline{E_2} \cap E_4] && (24)\\
&\leq n \cdot \mathbf{Adv}^{Bind}_{\mathcal{CS}}(B), && (25)
\end{aligned}$$

thus completing the proof.

□

5 Comparing RAONT-RS and HK2

We briefly introduce Krawczyk's SSMS scheme [9] and a robust extension, called HK2 [16]. We then compare AONT-RS with SSMS. This comparison can also be applied to the AONT-RS and HK2.

5.1 The SSMS and HK2 Scheme

Krawczyk's SSMS is a CSS [9]. It assumes an ind-1 secure encryption system, an IDA and a $(t, n)-$PSS. Intuitively, SSMS takes as input a message M, generates a key k and calculates $C \xleftarrow{\$} Enc_k(M)$. The ciphertext C is then shared amongst the n participants via an IDA whilst k is shared via a $(t, n)-$PSS. A player's share is one element of C and one of k. Krawczyk then extended his scheme to be robust in the RO model by using hash functions [9], which was proven to be secure in the RO model in [16].

HK2 is a robust extension of SSMS [16] using commitment schemes. HK2 relies on the same assumptions as SSMS, but additionally assumes a $(t, n)-$ECC and an $\epsilon(\dot{)}-$hiding commitment scheme $\mathcal{CS}$. Our extension of AONT-RS to RAONT-RS used similar techniques as in the extension of SSMS to HK2; the commitment scheme is added to SSMS and each player stores their share, along with a a decommittal and multiple fragments of committals. For a more detailed description, the reader is directed to [16].

We chose to use commitment schemes to extend AONT-RS to be robust (as was done in HK2) to achieve recoverability under standard assumptions. Instead, hash functions could be used, as in [9], to achieve recoverability in the RO model.

5.2 Comparison

In [14], Resch and Plank only briefly compare AONT-RS to Krawczyk's SSMS [9]. They then conduct a performance comparison of the AONT-RS with Rabin's IDA [12] and Shamir's PSS [17]. However, Rabin's IDA has no security requirements, and would not be used to distribute data if there were any privacy concerns, and Shamir's PSS achieves perfect security, which would not be used to share large data due to the bounds on the share sizes. Krawczyk's SSMS achieves computational security, which is a similar to AONT-RS. Thus we compare SSMS with AONT-RS. Similarly, RAONT-RS can be compared with HK2.

We compare the security, share size and efficiency of AONT-RS with SSMS. We exclude the contribution to the complexity made by $\mathcal{E}$, as this is equal in both schemes. The comparison is applicable to RAONT-RS and HK2, if we exclude the contribution to the share size and efficiency from $\mathcal{CS}$ (which is equal in both).

For the comparison, we will assume both AONT-RS and SSMS use the systematic RS IDA and that SSMS uses an ideal PSS. Such PSSs include Shamir's PSS [17], or Chen et al. [6]. Let $k \in \{0, 1\}^{\lambda}$.

Assume M is to be distributed and $\mathcal{E}$ is length preserving, so $|C| = |M|$. Let $C \in \{0,1\}^{\omega}$, and fix ω such that $\omega \geq \lambda(t-1)$. This is assumed to prevent attacks described in Section 3.2 against the AONT-RS, and to ensure all schemes are distributing a message of equal length, thereby allowing for a fair comparison. It is noted that if $\omega < \lambda(t-1)$, the AONT-RS will need to pad the message to lengthen C, whereas SSMS can distribute C as is. However, as mentioned previously, CSS schemes are often used when M is large, thus it is reasonable to assume that $\omega \geq \lambda(t-1)$. To illustrate, we highlight an example presented in [14]: they distribute a 4KB block of data using a 128 bit key amongst 16 servers such that any 10 can recover the data. So $n = 16, t = 10, \lambda = 128$ and $C \in \{0,1\}^{32000}$ with $\omega = 32000 >> \lambda(t-1) = 1152$.

5.2.1 Security.

AONT-RS achieves computational privacy, assuming $\mathcal{E}$ is ind-1 secure, H is indistinguishable from a RO and the IDA is equivalent to a $(0, t; n)-$linear ramp scheme. SSMS also assumes $\mathcal{E}$ is ind-1 secure and requires a $(t, n)-$PSS and an IDA (with no privacy requirements).

As SSMS is secure under standard assumptions, whereas AONT-RS is only secure in the RO model, SSMS is considered to be more secure.

5.2.2 Share Size.

The share given to each player from the AONT-RS is $\left\lceil \frac{\omega+\lambda}{t} \right\rceil$ bits. For SSMS, each share is $\left\lceil \frac{\omega}{t} \right\rceil + \lambda$ bits.

The AONT-RS achieves smaller share sizes than SSMS when $t \geq 2$ (which is true in general). The ratio between the share sizes is larger when t is bigger. The main contribution to the share size is from C, meaning the ratio between the share sizes will be small if ω is large and large if ω is small (meaning ω is close to $\lambda(t-1)$).

5.2.3 Efficiency of *Share*.

$Share^{\text{AONT}}$ requires one hash computation and $\mathcal{O}(\lambda(n+1)+n\omega)$ bitwise XORs (if multiplication is implemented via a look-up table).

In SSMS distribution of k via either [17] or [6] requires $\mathcal{O}(tn\lambda)$ bitwise XORs. The distribution of C via the IDA requires $\mathcal{O}(n\omega)$ XORs. Thus distribution of SSMS requires $\mathcal{O}(\lambda(tn) + n\omega)$ bitwise XORs.

AONT-RS requires fewer XORs than SSMS. For larger values of t, SSMS requires more XORs, whereas the complexity of AONT-RS is independent of t.

5.2.4 Efficiency of *Recover*.

Assume t players pool their shares. $Recover^{\text{AONT}}$ requires one hash function computation and $\mathcal{O}(t(\omega + \lambda) + \lambda)$ bitwise XORs.

SSMS requires $t(t-1)\left\lceil\frac{\omega}{t}\right\rceil$ bitwise XORs to recover C and either $\mathcal{O}(tn\lambda)$ (if [6] is the chosen PSS), or $\mathcal{O}(t\log^2 t\lambda)$ (for Shamir's PSS [17]) XORs to recover k. The total efficiency is the sum of the recovery of C and k.

Generally, the AONT-RS requires fewer bitwise XORs and is dependent only on t. Recovery of M using SSMS is dependent on the efficiency of the PSS used.

6 Conclusion

We generalised the AONT-RS and showed information is leaked when ciphertexts are shorter than $\lambda(t-1)$. We proved the AONT-RS scheme has computational privacy in the RO model. We extended the scheme to be robust and proved it achieves computational privacy in the RO model and recoverability under standard assumptions. Finally, we compared AONT-RS with SSMS, which is a comparison that can be used to compare RAONT-RS with HK2. We showed the (R)AONT-RS schemes achieve weaker security than SSMS/HK2 because their proofs are in the RO model, whereas SSMS/HK2 are provable under standard assumptions. However, by compromising security, (R)AONT-RS achieves smaller shares and more efficient dispersal and recovery.

References

[1] A. Beimel. Secret-sharing schemes: A survey. pages 11–46, 2011.

[2] G.R. Blakley. Safeguarding cryptographic keys. In *Proc. of the National Computer Conference1979*, volume 48, pages 313–317, 1979.

[3] V. Boyko. On the security properties of oaep as an all-or-nothing transform. In *Annual International Cryptology Conference*, pages 503–518. Springer, 1999.

[4] A. Chandrasekara, R. Bala, and G. Landers. Critical Capabilities for Object Storage. Technical report.

[5] H. Chen and R. Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *Annual International Cryptology Conference*, pages 521–536. Springer, 2006.

[6] L. Chen, T. M. Laing, and K. M. Martin. Efficient, "xor"-based, ideal (t, n)-threshold schemes. In *International Conference on Cryptology and Network Security*, pages 467–483. Springer, 2016.

[7] IBM. IBM Cloud Object Storage. `https://www.cleversafe.com/platform/why-ibm-cloud-object-storage`, 2016. Accessed: 2016-09-04.

[8] E. D. Karnin, J. W. Greene, and M. E Hellman. On secret sharing systems. *Information Theory, IEEE Transactions on*, 29(1):35–41, 1983.

[9] H. Krawczyk. Secret sharing made short. In *Annual International Cryptology Conference*, pages 136–146. Springer, 1994.

[10] F.J. MacWilliams and N.J.A. Sloane. *The theory of error correcting codes*. Elsevier, 1977.

[11] R.J. McEliece and D.V. Sarwate. On sharing secrets and reed-solomon codes. *Communications of the ACM*, 24(9):583–584, 1981.

[12] M.O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM (JACM)*, 36(2):335–348, 1989.

[13] I.S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[14] J. K. Resch and J. S. Plank. AONT-RS: blending security and performance in dispersed storage systems. In *FAST-2011: 9th Usenix Conference on File and Storage Technologies*, pages 191–202, February 2011.

[15] R. L. Rivest. All-or-nothing encryption and the package transform. In *International Workshop on Fast Software Encryption*, pages 210–218. Springer, 1997.

[16] P. Rogaway and M. Bellare. Robust computational secret sharing and a unified account of classical secret-sharing goals. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 172–184. ACM, 2007.

[17] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[18] M. Tompa and H. Woll. How to share a secret with cheaters. *Journal of Cryptology*, 1(3):133–138, 1989.