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

Liqun Chen[1, 2], Thalia M. Laing[3] and Keith M. Martin[3]

[1]Hewlett-Packard Enterprise, Bristol, UK; liqun.chen@hpe.com
[2]University of Surrey, Guildford, UK; liqun.chen@surrey.ac.uk
[3]Information Security Group, Royal Holloway, University of London, Egham, UK; {thalia.laing, keith.martin}@rhul.ac.uk

Abstract

In 2010, Resch and Plank proposed a computationally secure secret sharing scheme, called the AONT-RS scheme. We present a generalisation of their scheme and discuss two ways in which information is leaked if the scheme is used to distribute small ciphertexts. We then 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, ensuring the distributed data is recoverable even if a bounded number of players submit incorrect shares. We prove the robust AONT-RS scheme achieves computational privacy and recoverability in the random oracle model. Finally, we compare the security, share size and complexity of the robust AONT-RS scheme with a refined version of Krawczyk's robust scheme by Bellare and Rogaway.

1 Introduction

A threshold secret sharing scheme describes how to distribute data amongst multiple servers such that a threshold number can collaborate in order to uniquely reconstruct the data. The data maintains confidentiality if an insufficient number of servers collaborate.

Secret sharing is particularly useful in distributed storage systems where data is stored across multiple servers. Ideally, the data stored by each server, called their share, is smaller than the original data. A user wishing to access the data must access a threshold number of servers and then combine the retrieved shares.

A distributed storage system such as this enables greater availability, as there is no single point of failure, and adds redundancy to the system, thereby improving reliability in case of failures. Furthermore, the servers can be physically distributed, allowing for proximity to distributed clients, thereby improving both scalability and performance. Secret sharing also offers security without the reliance on cryptographic keys. In a threshold secret sharing scheme, an attacker must corrupt at least a threshold number of servers; this is in contrast to other encryption strategies, where the adversary need only corrupt the one location storing the cryptographic keys in order to gain access.

In 2010, Resch and Plank proposed a technique for a dispersed storage system, called AONT-RS [22], which blends an all-or-nothing transform (AONT) [23] with Reed-Solomon

(RS) coding [21]. The result is a computationally secure $(t, n)-$threshold secret sharing scheme in which data is distributed across n servers such that any t servers are able to recover the data and any set of $t-1$ servers reveals only a negligible amount of information. The AONT-RS scheme is a feature in the object storage system sold by Cleversafe, a company recently acquired by IBM [10], who renamed the product to IBM Cloud Object Storage. In 2010 there were over twenty AONT-RS dispersed storage installations around the world [22]. Since then, the system was rated the overall leader in the Gartner Critical Capabilities for Object Storage Report in 2016 [7] and its use is expected to grow. The scheme was developed to address the corruption, theft or loss of devices; it succeeds to the extent that if up to $t-1$ devices are lost the data will maintain confidentiality. However, if any of the n servers are corrupted and submit incorrect shares, the data is unrecoverable.

In this paper, we present a generalised version of the AONT-RS. In the original scheme, the block cipher mode of operation, the information dispersal algorithm (IDA) and a method of ensuring the integrity of the scheme were all specified and embedded in the AONT-RS definition. We generalise the scheme for any block cipher mode of operation and any IDA satisfying previously undefined security properties. This gives users flexibility to utilise IDAs already implemented in their systems. We also define the (previously undefined) security properties required by the internal encryption system.

We then discuss how the AONT-RS scheme leaks information if it is used to distribute a small (relative to the security parameter and threshold value) ciphertext. We illustrate two ways this can happen and discuss what size the data must be in order to avoid this.

Resch and Plank claim their scheme has integrity due to the use of a canary, which enables an authorised user to confirm whether or not the correct data has been recovered. However, if the incorrect data was recovered due to an incorrect share being submitted, the user cannot recover the correct data. To address this issue, we extend the AONT-RS scheme to be robust by using commitment schemes, as in [24]. The robust scheme ensures that, even if a bounded number of false shares are submitted, the original data will be recovered.

Resch and Plank also claim their scheme achieves computational security but no thorough security analysis is provided. We prove that both the robust and non-robust AONT-RS schemes achieve computational privacy in the random oracle (RO) model. We then prove the robust AONT-RS achieves computational recoverability, also in the RO model.

Resch and Plank compare the performance of their scheme to Shamir's perfect secret sharing scheme [25] and Rabin's IDA [20]. However, both these schemes have different security properties to the AONT-RS, with Shamir's scheme achieving perfect security and Rabin's IDA achieving incremental security, where each share reveals a measured amount of information about the data. Shamir's scheme achieves stronger security than the AONT-RS, which in turn achieves stronger security than Rabin's IDA. Here, we compare the AONT-RS scheme to Krawczky's secret sharing made short (SSMS) scheme [13] and the robust AONT-RS to a robust extension of SSMS by Bellare and Roagway, called HK2 [24]. We choose these schemes because they provide computational security. The SSMS scheme was the first computationally secure threshold secret sharing scheme to be proposed and is the most well-known and used scheme. Resch and Plank mention the SSMS scheme, but only briefly in an example comparing the share size and security. No further comparison (such as performance) is conducted.

We compare the security and share size of our robust AONT-RS scheme and HK2. We then compare the number of bitwise XORs required to distribute and recover data via both schemes. During the comparison, we do not note the contribution of the commitment schemes to the share size or complexity, as this contribution would be identical for each

scheme. As a result, the comparison is also applicable to the non-robust AONT-RS and SSMS [13].

Our comparison shows the AONT-RS scheme achieves weaker security than the SSMS scheme; similarly, the robust AONT-RS achieves weaker security than HK2. This is because, when proving the robust and non-robust AONT-RS, we are forced to use the RO model due to the use of a hash function. In contrast, SSMS and HK2 are provably secure under standard assumptions. However, in reducing the security achieved, the (robust) AONT-RS scheme achieves smaller share sizes, especially when either the security parameter or the threshold value is large, and can be implemented using fewer bit-wise XORs, but requires a call to a hash function, which the HK2 scheme does not.

1.1 Related Work

Shamir and Blakely independently introduced secret sharing schemes in 1979 [25, 4]. Shamir's was a threshold scheme that achieved information security and minimum share sizes.

In 1994, Krawczyk published a paper proposing a computationally secure secret sharing scheme (CSS) in the non-robust setting, which the SSMS scheme [13]. Also in this paper, Krawczky proposed goals for a robust CSS scheme, along with a candidate solution. Previously, the CSS goal had been mentioned by Karnin et al. [11], along with a variant considering the detection, but not correction, of cheating. Prior to Krawczyk's work, robustness had only be studied in the information-theoretic setting by McEliece and Sarwate [19] and Tompa and Woll [28].

The motivation to Krawczyk's work was to achieve shares smaller than were possible in information theoretically secure secret sharing schemes [11]. Krawczyk utilised an IDA proposed by Rabin [20] in order to achieve smaller share sizes, then added on robustness by utilising a hash function technique proposed previously [14].

Much of the follow-on work to Krawczyk's paper focused on achieving CSS for generalised access structures (rather than for threshold schemes), such as in [1, 6, 18, 29]. Krawczyk's work was revisited in 2007 by Bellare and Rogaway [24], in which they revisited the basics of the robust CSS, proposed formal definitions 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) that 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 [22], which is studied in detail here and utilises all or nothing transforms and IDAs.

1.2 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 the AONT-RS scheme that occur if the data being dispersed is small in relation to the security parameter and threshold value. We discuss what size the data must be to prevent this leakage.
- We prove the AONT-RS scheme achieves computational privacy in the RO model.
- We extend the AONT-RS scheme to be robust using commitment schemes and proof the robust AONT-RS scheme achieves computational privacy and recoverability in the RO model.

- Finally, we conduct a comparison of the robust AONT-RS scheme with the HK2 scheme. We compare the security, storage required and the difference in the bitwise XOR complexities for both distribution and recovery of the secret. We do not include the contribution to the share size from the commitment scheme and the contribution to the complexity from both the commitment scheme and the encryption scheme. This is because the contribution to both the robust AONT-RS and HK2 would be identical. Therefore, our comparison is also applicable to the generalised, non-robust AONT-RS scheme and Krawczyk's SSMS.

1.3 Organisation

This paper is organised as follows. In Section 2 we present notation and provide definitions. In Section 3 we present a generalised version of the AONT-RS scheme and discuss information leakage when the ciphertext is small, relative to the security parameter and threshold value. We then prove the AONT-RS scheme achieves computational privacy in the RO model. In Section 4 we extend the AONT-RS scheme to be robust and prove our extended scheme achieves both computational privacy and recoverability also in the RO model. In Section 5 we introduce the HK2 scheme and compare it with the robust AONT-RS scheme. Our comparison considers the security and share size of the schemes, as well as the number of bitwise XORs required to distribute and recover data. 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: $\Pi = \{Share, Recover\}$. $Share$ is a probabilistic algorithm that takes as input a* secret *s chosen from* secret space *$\mathcal{S}$ and outputs an $n-$vector $\boldsymbol{S}$. Each player P_i receives a* share *$\boldsymbol{S}[i]$ for $1 \leq i \leq n$. $Recover$ is a deterministic algorithm that takes as input shares from players in $\mathcal{P}$ and outputs some value $s' \in \{\mathcal{S} \cup \perp\}$. The secret should be* recoverable *by any set of at least t players who submit their shares to $Recover$, and* private*, so any set of fewer than t players (called* unauthorised sets*) are unable to recover s (meaning that $s' \neq s$).*

A $(t, n)-$secret sharing scheme can either have *perfect* or *computational* security. A perfectly secure scheme requires unauthorised sets to learn no information about s, whereas in a computationally secure scheme a negligible amount of information about s can be learnt.

Both perfectly and computationally secure schemes can be defined by security games between an adversary and a challenger: one game defines the privacy of the scheme and the other the recoverability. Both games are attributed to Bellare and Rogaway [24]. 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.

The privacy game $Priv$ is described in Figure 1. Given the parameters t and n, the challenger chooses a bit b at random. The adversary then chooses two secrets $s_0, s_1 \in \mathcal{S}$ and sends these to the challenger. The challenger checks that $s_0, s_1 \in \mathcal{S}$; if they are not, the challenger halts and returns $\perp$ to the adversary. Otherwise, the challenger inputs s_b to

GAME *Priv*

- PROCEDURE $Initialise(t, n)$
  $b \xleftarrow{\$} \{0,1\}$; $j = 1$
- 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 $\boldsymbol{S}[i]$; $j = j+1$
  Else halt.
- PROCEDURE $Finalise(b')$
  Return $b' = b$

GAME *Rec*

- PROCEDURE $Initialise(t, n)$
  $T \leftarrow \emptyset$; $j = 1$
- 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 $\boldsymbol{S}[i]$
  $j = j+1$; $T \leftarrow T \cup \{i\}$
  Else halt.
- PROCEDURE $Finalise(\boldsymbol{S}_T)$
  Return $s \neq s' \leftarrow Recover(\boldsymbol{S}_T \cup \boldsymbol{S}_{\overline{T}})$

Figure 1: Games used to define privacy and recoverability of a $(t, n)-$secret sharing scheme

the *Share* algorithm, which outputs the $n-$vector $\boldsymbol{S}$. Nothing is returned to the adversary at this stage. The adversary can then make up to $t-1$ queries of the form $Corrupt(i)$ for $1 \leq i \leq n$ and receives the corresponding share $\boldsymbol{S}[i]$ in return. After the corrupt stage, the adversary must output a guess b' for b. The adversary 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 the finalise procedure. Define the *advantage* of A as

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

The recoverability game *Rec* for a $(t, n)-$secret sharing scheme is also defined in Figure 1. Intuitively, the game models an adversary's ability to prevent the recovery of s by either deleting shares or submitting false shares to the *Recover* algorithm. The game is initialised by letting the set T, which will denote the players the adversary corrupts, be the empty set. The adversary then chooses a secret s and submits this to the challenger, who inputs it to the *Share* algorithm. As before, nothing is returned to the adversary at this stage. The adversary can then make up to $n-t$ queries of the form $Corrupt(i)$ for $1 \leq i \leq n$ and receives the corresponding share $\boldsymbol{S}[i]$ in return. Each party corrupted by the adversary is noted in the set T. During the finalise procedure, the adversary outputs a partially complete $n-$vector $\boldsymbol{S}_T$, consisting of the altered (or deleted) shares queried during the corrupt procedure. This vector is then completed by the challenger by filling the remaining elements with valid shares from uncorrupted players; these shares are noted in the vector $\boldsymbol{S}_{\overline{T}}$. The complete vector $\boldsymbol{S}_T \cup \boldsymbol{S}_{\overline{T}}$ is then submitted to the *Recover* algorithm. The adversary wins if the recovered secret s' is not equal to s.

The recoverability game allows the adversary to call up to $n-t$ shares during the *Corrupt* procedure. However, of these $n-t$ shares, the adversary is allowed to to replace at most t with strings; the rest must be replaced by an empty string [24].

Let A be an adversary playing the recoverability game *Rec* against Π. Call A a *recoverability adversary*. Let $\Pr[Rec^A]$ denote the probability s is not recovered by the deterministic algorithm $Recover(\boldsymbol{S}_T \cup \boldsymbol{S}_{\overline{T}})$. Define the advantage of A as

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

The games *Priv* and *Rec* in Figure 1 can be used to define perfectly secure and computationally secure $(t, n)-$secret sharing schemes.

Definition 2 *A* perfectly secure $(t, n)-$secret sharing scheme (PSS) *is a* $(t, n)-$*secret sharing scheme in which a privacy adversary has an advantage of* 0 *and a recoverability adversary who is restricted to only deleting (and not corrupting) shares has an advantage of* 0*.*

Let $Share^{PSS}$ and $Recover^{PSS}$ denote the distribution and recovery algorithms of a PSS. Intuitively, in a PSS, a privacy adversary has no advantage over guessing when outputting a guess b'. In a PSS, the size of the share $\boldsymbol{S}[i]$ given to each player P_i must be at least the size of the secret s [2]; schemes in which this bound are met are called *ideal*. This bound can, however, be problematic. In settings where the secret is large or the storage available to each player is small, it may be preferable to use a scheme achieving computational, rather than perfect, security as this allows for smaller share sizes.

Definition 3 *A* computationally secure $(t, n)-$secret sharing scheme (CSS) *is a* $(t, n)-$*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* 0*.*

In a CSS, the advantage of the privacy adversary is negligible, but greater than zero. Computationally secure schemes are less secure than perfectly secure schemes but are able to achieve smaller share sizes. In general, computationally secure schemes are sufficient for most applications [13].

In both PSS and CSS schemes, the recoverability adversary is limited to deleting, and not altering, shares. It is guaranteed that, as long as t shares are submitted, s will be recovered. A robust scheme ensures the recovery of the secret in the setting where 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.*

Another way of constructing schemes with share sizes smaller than a PSS is by further relaxing the security requirement so sets of fewer than t players learn a measured amount of information about the secret. *Ramp schemes* achieve this and can be defined using information theoretic notation [2]. Let $\boldsymbol{\mathcal{S}}$ denote the discrete random variable corresponding to the choice of secret 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 *is a method of distributing a secret* k *such that any set of at least* t_1 *players can pool their shares to uniquely recover the secret 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, any set of at least t_1 players can pool their shares to uniquely recover the secret. A set of t_0 or fewer players reveals no information about the secret. 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. In fact, after t_0 shares are pooled, every further share reveals $\frac{1}{t_1 - t_0}$ bits of information about s.

Observe that a $(t, n)-$PSS is a $(t-1, t; n)-$ramp scheme.

2.2 Symmetric Key Encryption

We now consider the definition and security of a symmetric key encryption scheme [12].

Definition 6 *A* symmetric key encryption scheme $\mathcal{E}$ *consists of a message space* $\mathcal{M}$*, a keyspace* $\mathcal{K} \subseteq \{0,1\}^\lambda$ *with security parameter* λ*, a ciphertext space* $\mathcal{C}$*, and three algorithms:*

- *The key-generation algorithm* $KeyGen$ *is a probabilistic algorithm that outputs a key* k *chosen uniformly at random from* $\mathcal{K}$*.*
- *The encryption algorithm* Enc *takes as input a key* $k \in \mathcal{K}$ *and a plaintext message* $M \in \mathcal{M}$ *and outputs a ciphertext* $C \in \mathcal{C}$*. Denote* $Enc_k(M)$ *as the encryption of the message* M *under the key* k*.*
- *The decryption algorithm* Dec *takes as input a key* $k \in \mathcal{K}$ *and a ciphertext* $C \in \mathcal{C}$ *and outputs a plaintext* $M \in \mathcal{M}$*. Denote* $Dec_k(C)$ *as the decryption of the ciphertext* C *under the key* k*.*

A symmetric key encryption scheme $\mathcal{E} = (\mathcal{M}, \mathcal{K}, \mathcal{C}, KenGen, Enc, Dec)$ must satisfy the *correctness requirement*: for every key $k \in \mathcal{K}$ and $M \in \mathcal{M}$,

$$Dec_k(Enc_k(M)) = M. \tag{4}$$

That is, encrypting a plaintext message M under k and then decrypting the resulting ciphertext C under the same key k must result in the return of the original plaintext M.

Block ciphers (such as AES [9]) are symmetric encryption schemes that operate on blocks of plaintext, rather than individual bits, and may use different modes of operation [27]. These modes enable randomisation of the plaintext, normally by using an *initialisation vector*, to give *probabilistic encryption*. To denote an encryption algorithm using randomisation, we use the notation $C \xleftarrow{\$} Enc_k(M)$.

We now define the notion of indistinguishability in an encryption scheme $\mathcal{E}$ in Figure 2. Intuitively, the challenger randomly chooses a bit b and generates an encryption key $k \in \mathcal{K}$. The adversary submits two distinct messages $M_0, M_1 \in \mathcal{M}$ of equal length. The challenger encrypts M_b and returns the ciphertext C to the adversary. The adversary may repeat this procedure multiple times. The adversary then outputs a guess b' for b and wins if $b' = b$.

Let A be an adversary playing the indistinguishability game Ind, as in Figure 2, against $\mathcal{E}$. Call A an *indistinguishability adversary*. Let $\Pr[Ind^A]$ denote the probability A outputs the correct guess b' for b during the finalise procedure. Define the advantage of A as

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

We say the encryption scheme $\mathcal{E}$ has the property of indistinguishability if the advantage of A is negligible.

In Game Ind, the adversary is allowed to repeat the deal procedure multiple times. We can, however, weaken the adversary and limit them to only calling the deal procedure once. We call an adversary bounded in this way an *ind-1 adversary*. The advantage of an ind-1 adversary is computed in the same way as an indistinguishability adversary, as in (5). Any scheme that achieves indistinguishability also achieves ind-1 security, as an ind-1 adversary is weaker than a general indistinguishability adversary.

GAME *Ind*

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

Figure 2: Game defining indistinguishability of a symmetric key encryption scheme $\mathcal{E}$

2.3 Commitment Schemes

A *commitment scheme* $\mathcal{CS}$ is a triple of algorithms $(ParGen, Ct, Vf)$ run between a sender and a receiver. The *sender* wishes to commit to a message M chosen from some message space $\mathcal{M}$. The message will be revealed to the receiver in the future but, until then, the sender wishes to keep M private. In the meantime, the sender wishes to provide the receiver with some form of guarantee that the M to be revealed is the same M the sender initially chose. This is achieved as follows.

The parameter generation algorithm $ParGen$ generates the public parameters π. The sender then runs the probabilistic commitment generation algorithm Ct using the public parameters π and the message M they wish to commit to as input. A *committal* H and *decommittal* R are then output. The sender sends the committal H to the receiver and keeps R and M private. When the sender wishes to reveal M, they must send M and R to the receiver, who will input all three values H, M and R to the deterministic verification algorithm Vf. A bit is then output: '1' if the verification was a success and the committal and decommittal were generated from M, or a '0' otherwise.

A commitment scheme should satisfy two security properties:

1. *Hiding.* The receiver should learn nothing about M from the commitment H.
2. *Binding.* It should be difficult to find H, M_0, M_1, R_0, R_1 such that $M_0 \neq M_1$, but $Vf(\pi, H, M_0, R_0) = Vf(\pi, H, M_1, R_1) = 1$.

The hiding property ensures the receiver has no advantage in guessing M given H, thereby ensuring the privacy of M for the sender. The binding property ensures the sender cannot cheat and send a decommittal $R_1 \neq R_0$ verifying a different message $M_1 \neq M_0$. The hiding and binding properties of a commitment scheme can be described using adversarial games, as in Figure 3.

Intuitively, the game $Hide$ is as follows. The challenger generates public parameters π and randomly chooses a bit b. In the deal procedure, the adversary chooses two messages $M_0, M_1 \in \mathcal{M}$ and sends these to the challenger. The challenger computes the committal H and decommittal R on the message M_b and parameters π. The challenger sends the committal H to the adversary. The adversary is allowed to repeat the deal procedure multiple times, but must finally submit a guess b' for b to the challenger and wins if $b' = b$.

Let A be an adversary playing the hiding game $Hide$, as in Figure 3, against $\mathcal{CS}$. Call A a *hiding adversary* and let $\Pr[Hide^A]$ be the probability A correctly guesses $b' = b$. Define the advantage of A against $\mathcal{CS}$ to be

$$\mathbf{Adv}^{Hide}_{\mathcal{CS}}(A) = 2 \cdot \Pr[Hide^A] - 1. \tag{6}$$

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 used to define the hiding and binding security properties of a commitment scheme $\mathcal{CS}$.

Say the commitment scheme $\mathcal{CS}$ 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.

The binding property of $\mathcal{CS}$ is defined in Figure 3. Intuitively, an adversary A wins if they are able to submit a message M_1 and corresponding decommittal R_1 that passes the verification algorithm for a committal token H, generated from a different message $M_0 \neq M_1$. The game begins with the adversary submitting a message M_0 to the challenger. If $M_0 \in \mathcal{M}$, the adversary will compute the committal H and decommittal R_0, and return these to the adversary. The adversary must then submit a message $M_1 \neq M_0$ and decommittal R_1, such that $Vf(H, M_1, R_1) = 1$.

Let A be a *binding adversary* playing the *Bind* game against $\mathcal{CS}$. Let $\Pr[Bind^A]$ be the probability A outputs a successful (M_1, R_1). The advantage of A is

$$\mathbf{Adv}^{bind}_{\mathcal{CS}}(A) = \Pr[Bind^A]. \tag{7}$$

2.4 Error Correcting Codes

An error correcting code (ECC) 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 [17].

Definition 7 *An* error correcting code *(ECC) E of* length n *over a finite alphabet F is a subset of F^n. The elements of E are called* codewords. *The* size *of E is $|E| = m$. The* minimum distance *of E is the minimum Hamming distance between any two distinct codewords and is denoted by d.*

Let E be a code of length n. We say E is *linear* if for all $u, w \in E$, we have $u + w \in E$, where addition is modulo q with $|F| = q$. Intuitively, a code is linear if all linear combinations of the codewords are also codewords. If $u_1, \ldots, u_t$ is a basis for a linear code E, then say E has *dimension* t. There are q^t possible codewords. Let d be the minimum distance of E. We say that E is an $[n, t, d]-$*code*.

One important ECC is a *maximum distance separable* (MDS) code [17], which is a linear code that meets the Singleton bound: $d = n - t + 1$ [26]. 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 [21] 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 to a codeword $n-$vector $\boldsymbol{U}$ via a $(t,n)-$ECC. The word u is recoverable from any t of the n elements in $\boldsymbol{U}$ via the deterministic algorithm $u \leftarrow Recover^{ECC}(\boldsymbol{U})$.

2.5 Information Dispersal Algorithms

Information dispersal was first introduced by Rabin [20].

Definition 8 *Let $t, n \in \mathbb{N}, t \leq n$. A $(t,n)-$*information dispersal algorithm *(denoted 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 will output the original message M.*

Intuitively, a $(t,n)-$IDA shares data between n players such that any set of at least t players can recover the data. This is equivalent to the recoverability property of a $(t,n)-$secret sharing scheme in which a recoverability adversary has an advantage of 0 in winning the *Rec* game. There are, however, no privacy requirements on an IDA. Therefore a trivial example of an IDA is replication: each player could be given a copy of M. This satisfies the requirement that any t of the n players could recover M but there is no privacy of M as any player could individually recover M.

Any $(t,n)-$secret sharing scheme also satisfies the conditions of an IDA as the secret is recoverable by t players. A $(t,n)-$secret sharing scheme has the additional property of privacy. However, it is possible to achieve smaller share sizes than in a $(t,n)-$secret sharing scheme by taking advantage of the lack of security requirements.

2.5.1 Resch and Plank's IDA.

In the AONT-RS scheme, Resch and Plank specify a systematic IDA to be used which is a variant of an RS code [22]. This IDA is of particular interest to us and we will henceforth refer to their IDA as the systematic RS-IDA.

A brief overview of the two algorithms $Share^{RS-IDA}$ and $Recover^{RS-IDA}$ constituting the systematic RS-IDA is given here.

$Share^{RS-IDA}$ is a probabilistic algorithm that takes as input a message M to be distributed between n players. Let $F = GF(2^{\omega})$ be a Galois field of characteristic 2. The message M is parsed into t words and treated as a $t-$vector, $\boldsymbol{M} \in F^t$. This vector is then multiplied on the left by an $n \times t$ binary matrix G, where multiplication of elements $b \in \{0,1\}$ and $d \in F$ is defined as follows: $\{0,1\} \times F \rightarrow F$, where $0 \times d = 0 \in F$ and $1 \times d = d \in F$. 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 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 new $t-$vector $\boldsymbol{V'}$ is created from these shares. A $t \times t$ matrix G' is then formed, consisting of the t rows of G corresponding to the shares pooled. This matrix is inverted and multiplied by the vector $\boldsymbol{V'}$ to return $(G')^{-1} \cdot \boldsymbol{V'} = \boldsymbol{M}$, from which M can be recovered.

In general, the $n \times t$ matrix G can be any binary matrix such that any t of the n rows are linearly independent. Resch and Plank chose to let the first t rows form the $t \times t$ identity

matrix to make the IDA systematic and thus more efficient as only the final $n-t$ elements of $\boldsymbol{V}$ need be encoded. The first t elements can be directly copied from the vector $\boldsymbol{M}$.

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 [8]. Thus the systematic RS-IDA used by Resch and Plank is equivalent to a $(0, t; n)-$linear ramp scheme, as in Definition 5. Every share pooled reveals 2^{ω} bits of information about the message M.

3 The AONT-RS

In this section, we consider the AONT-RS scheme proposed by Resch and Plank [22]. We present a generalised version of the scheme, and then discuss how the scheme leaks information when used to distribute ciphertexts that are small (relative to the security parameter and threshold value) and show how this can be prevented. Finally, we present a proof that the AONT-RS scheme achieves computational privacy in the RO model.

3.1 Generalising the AONT-RS

Resch and Plank propose a new CSS in [22], which they call the AONT-RS scheme; it is called this because 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 [5].

Resch and Plank assume the existence of a symmetric key encryption scheme $\mathcal{E}$ operating on blocks of plaintext, a cryptographic hash function H and the systematic RS-IDA, as in Section 2.5.1. They also assume the digest h of the chosen cryptographic hash function H is of equal length to the key k generated by $KeyGen$ from $\mathcal{E}$. They do not define what security properties the encryption scheme $\mathcal{E}$ must have.

For our generalised version, we also assume the existence of $\mathcal{E}$, which we observe requires ind-1 security, and H, but specify the use of any IDA with algorithms $Share^{IDA}$ and $Recover^{IDA}$, such that the IDA is equivalent to a $(0, t; n)-$linear ramp scheme (which the specified systematic RS-IDA is). We present our generalised version of the scheme, then discuss the generalisations made. For now on, let Π denote our generalised version of the AONT-RS scheme which consists of two algorithms, $\Pi = \{Share^{AONT}, Recover^{AONT}\}$. These algorithms have been summarised in Figure 4.

PROCEDURE $Share^{AONT}(M)$
1. $k \xleftarrow{\$} KeyGen(\{0,1\}^{\lambda})$; $\quad C \xleftarrow{\$} Enc_k(M)$
2. $h = H(C)$; $\quad c_d = h \oplus k$
3. $\boldsymbol{V} \leftarrow Share^{IDA}(C||c_d)$
4. 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 $M \in \mathcal{M}$, the $Share^{AONT}$ algorithm generates a key k of length λ, encrypts M under k, then computes the hash of the ciphertext $h = H(C)$. The digest of the hash function h is then XORed with the encryption key k to give a value c_d, which we will call the *difference value*. The ciphertext and difference value are then concatenated and dispersed via an IDA amongst the n players.

In order to fully recover the message M, at least t players must pool their shares into a vector $\boldsymbol{V}$. Using the algorithm $Recover^{IDA}$, C and c_d can be recovered. The digest $h = H(C)$ can then be calculated and XORed with c_d to recover k. The ciphertext C is then decrypted using k and M is recovered.

In the original proposition of the scheme, Resch and Plank specified that the encryption scheme be used in CBC mode [27] (but do not specify what security $\mathcal{E}$ requires), that the IDA used be the systematic RS-IDA, as in Section 2.5.1, and defined the use of a *canary* to provide integrity.

We have generalised the use of the encryption scheme $\mathcal{E}$ by not specifying the mode of operation to be used, but rather specifying the scheme $\mathcal{E}$ be probabilistic. 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. We have also identified that the scheme needs to have ind-1 security. This is because each time a new message M is shared, a new encryption key k is generated. So each key is only used to encrypt one message.

We generalised the IDA by allowing the use of any IDA equivalent to a $(0, t; n)-$linear ramp scheme. If the IDA is not equivalent to a $(0, t; n)-$linear ramp scheme and is, instead, any IDA, there are no privacy guarantees. For example, in Section 2.5, we discussed how replication satisfies the requirements of an IDA. However, using replication in the AONT-RS scheme would be disastrous as any player would know all of C and c_d and so would be able to compute k and recover M. A $(0, t; n)-$linear ramp scheme ensures players learn information about only the share they are given or those contributed to $Recover$.

Finally, Resch and Plank use a *canary* to provide integrity. The canary is a known, fixed value that is concatenated with the plaintext M. The idea is that when a sufficient number of players have pooled their shares, they can run the $Recover^{AONT}$ algorithm to recover some data M'. As the canary is a known value, they can check whether the canary is present in M'; if it is, then M' is verified and accepted as the original data $M = M'$. If not, it is concluded that one of the shares submitted to $Recover^{AONT}$ was incorrect.

In our generalisation of the AONT-RS scheme we have removed the concept of the canary. This is because, although the canary is able to detect whether or not the correct M has been recovered, if there has been a corruption there is no way for the scheme to correctly recover M. Our extension of the scheme to be robust in Section 4 is an improvement on the canary because it is able to correctly recover M, even if a number of players submit incorrect shares, as long as at least t shares are submitted correctly.

For now, we will focus on the non-robust version of the scheme and discuss how it can leak information when a small ciphertext, relative to the security parameter λ and the threshold value t, is distributed. This leakage, along with how to avoid it is discussed next.

3.2 Information Leakage

Resch and Plank claim their system is secure because, if only $t-1$ players pool their shares, the entirety of $\boldsymbol{V}$ will not be recoverable, due to the security properties of the systematic RS-IDA (which is equivalent to a $(0, t; n)-$ramp scheme). Without recovering the entirety of $\boldsymbol{V}$, the $t-1$ players are unable to learn both C (which is needed to compute $h = H(C)$) and c_d. Resch and Plank assume that, at most, the adversary will be able to learn either

- some or all of c_d and some of (but not all of) C, or
- none of c_d and all of C.

Note that C is encrypted, so it does not affect the security of the system if the adversary learns some of (or all of) C. The security of the scheme is affected however, if the adversary

is able to learn all of C and some of c_d, or all of c_d and a sufficient amount of C, which may then enable them to learn information about the plaintext.

We will show that, in the case when C is a short ciphertext (in relation to the security parameter λ and the threshold value t), the adversary may learn all of C and some of c_d, resulting in learning partial information about the key k and thereby reducing the security of the system. Then, we will show a situation in which the adversary learns all of c_d and only partial information about C, but may be able to recover the key k.

3.2.1 Learning C completely and partial information about c_d.

To illustrate the leakage of information for a small ciphertext C, consider the following example. Let C and k both be strings of 128 bits. So $C||c_d$ totals 256 bits. Assume the IDA used in the systematic RS-IDA as in Section 2.5.1, which is equivalent to a $(0, t; n)-$linear ramp scheme. Let $\boldsymbol{M}$ denote the $t-$vector to be distributed via the IDA, which is formed from $C||c_d$. Let there be $n = 5$ players $\mathcal{P} = \{P_1, \ldots, P_5\}$ and let the threshold $t = 4$. The string $C||c_d$ would be split into four words to make up 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, so $C_0, C_1, c_{0,d}, c_{1,d} \in \{0,1\}^{64}$. This $4-$vector would then be multiplied on the left by the generator matrix $G \in \{0,1\}^{(5\times 4)}$, which would give

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

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 of players, yet they would be able to combine their shares to learn all of C and half of c_d. They could then correctly compute $H(C) = h$ and XOR the first half of h with the known first half of c_d to find the first half of the key k. This would reduce the security of the system from 128 bits to 64 bits.

This attack can be prevented if c_d is contained entirely in one element of $\boldsymbol{M}$. This ensures that a share cannot leak partial information of c_d and, instead, is learnt 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 partial information about C and c_d completely.

An alternative version of this attack can be conducted when c_d is known completely and only partial information of C is known. Resch and Plank claim that, due to the hash function H, all of C must be learnt in order to calculate h. However, this is not necessarily true as, even in the *random oracle model* (RO) [3] where each call to the hash function H is treated as a call to a random oracle, the hash function is deterministic.

Consider the following example. Assume an attacker is able to learn all of c_d but only partial information of C. If an adversary knows all but (for example) one bit of C, they can construct the two possibilities for C (one where the unknown bit is a '0', label this possibility C_0 and the other when it is a '1', label as C_1) and compute the two corresponding hashes, $h_0 = H(C_0)$ and $h_1 = H(C_1)$ respectively. The adversary can then compute two key

PROCEDURE *Initialise* G_0

$k \xleftarrow{\$} \{0,1\}^\lambda$; $b \xleftarrow{\$} \{0,1\}$; $k' = k$

PROCEDURE $Deal(x_0, x_1)$ G_0, 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 *Initialise* G_5

$k, k' \xleftarrow{\$} \{0,1\}^\lambda$; $b \xleftarrow{\$} \{0,1\}$

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 $(b' = b)$

Figure 5: Games for proving Theorem 1, the privacy of the AONT-RS scheme.

candidates $k_0 = c_d \oplus h_0$ and $k_1 = c_d \oplus h_1$ and decrypt the two ciphertexts C_0 and C_1 with the corresponding candidate keys and reveal two plaintext messages M_0 and M_1. From these, the adversary can guess which plaintext message is likely to be the true plaintext message and has thus learnt k.

In general, if the adversary knows c_d and all but i bits of C, the adversary can conduct this attack. If $i < \lambda$, where $k \in \{0,1\}^\lambda$, this attack is quicker than a brute force attack. Therefore, every adversary must be unable to learn at least λ bits of C if c_d is known. In order to ensure this, each element in $\boldsymbol{M}$ must be at least the size of the key. This is true if $C \in \{0,1\}^\omega$, where $\omega \geq (t-1)\lambda$.

In general, both attacks can be prevented if $k \in \{0,1\}^\lambda$ and $C \in \{0,1\}^\omega$, where $\omega \geq (t-1)\lambda$. This forces c_d to be contained entirely in one share meaning that any unauthorised set of players will learn either no information about c_d, or be missing at least λ bits of information about C. 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 of the AONT-RS, we must make additional assumptions on the security of the cryptographic hash function and the encryption system, as discussed in the next section.

3.3 Proving the Privacy of AONT-RS

In this section, we 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 and let the internal hash function H be a random oracle. Then there is an ind-1 adversary B attacking the indistinguishability of $\mathcal{E}$ such that*

$$\mathbf{Adv}^{Priv}_{\Pi}(A) \leq \mathbf{Adv}^{Ind}_{\mathcal{E}}(B), \tag{8}$$

where B makes only one query during the deal procedure of Game Ind (as in Figure 2) and the running time of B is that of A plus overhead consisting of one execution of the $Share^{AONT}$ algorithm of Π.

The proof relies on games G_0 and G_5, as defined in Figure 5. The advantage of the AONT-RS privacy adversary A can be defined as

$$\mathbf{Adv}^{Priv}_{\Pi}(A) = 2 \cdot \Pr[G^A_0] - 1. \tag{9}$$

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

$$\Pr[G_0^A] = \Pr[G_5^A], \tag{10}$$

by utilising the assumption that the hash function H behaves as a random oracle. Because of the IDA used to distribute $(C||c_d)$ and the restriction that $C = \{0,1\}^\omega$, where $\omega \geq \lambda(t-1)$, the adversary is always missing at least λ bits of C or all of c_d. Thus the adversary cannot learn either h or c_d, but can learn the other. If A knows h, then $h = c'_d \oplus k'$. If A knows c_d, then $c_d = h' \oplus k'$, for some $c'_d \neq c_d$ and $h' \neq h$. Thus (10) holds true.

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

$$2 \cdot \Pr[G_5^A] - 1 \leq \mathbf{Adv}_{\mathcal{E}}^{Ind}(B). \tag{11}$$

Adversary B picks $k' \xleftarrow{\$} \{0,1\}^\lambda$ at random and runs A. Adversary A submits $Deal(x_0, x_1)$, B then 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 game G_5 by 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 a $Corrupt(i)$ query, B can respond with the value $\boldsymbol{V}[i]$. When A halts and outputs a bit b', adversary B passes this onto their challenger as their guess. The advantage of B is $2 \cdot \Pr[b' = b] - 1$.

By combining (9), (10) and (11), we see that

$$\mathbf{Adv}_{\Pi}^{Priv}(A) \leq \mathbf{Adv}_{\mathcal{E}}^{Ind}(B), \tag{12}$$

as required.

□

4 Extending AONT-RS to be Robust

In [24], Bellare and Rogaway extend Krawczyk's SSMS [13] to be robust by using commitment schemes. This technique can be applied make the AONT-RS robust.

Let $\mathcal{E}$ be an encryption scheme with ind-1 security and assume the existence of a $(t,n)-$ECC with distribution and recovery algorithms $Share^{ECC}$ and $Recover^{ECC}$ and an IDA equivalent to a $(0,t;n)-$linear ramp scheme. Let Ct and Vf be algorithms in a commitment scheme $\mathcal{CS}$ that achieve the hiding and binding property. Let H be a hash function.

Let $\Pi_R = \{Share^{RAONT}, Recover^{RAONT}\}$ denote the robust AONT-RS scheme, where $Share^{RAONT}$ and $Recover^{RAONT}$ are defined as in Figure 6. Intuitively, the scheme is the same as the AONT-RS scheme defined in Figure 4. However, in addition to being given the share $\boldsymbol{V}[i]$, each player is also 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 of the committal $\boldsymbol{H}[i]$ dispersed via a $(t,n)-$ECC . Let $\boldsymbol{S}_i[j]$ be the j^{th} element of $\boldsymbol{S}_i$.

Let $\Diamond$ denote an empty share (one that has not be submitted). As long as t players submit uncorrupted shares, $Recover^{RAONT}$ will recover the committal values $\boldsymbol{H}[i]$, which can then verify each player's share and highlight any corrupted shares.

This technique is an improvement on the use of a canary, suggested in the original AONT-RS scheme [22]. Unlike a canary, the use of a commitment scheme means that, even

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

1. $k \xleftarrow{\$} \{0,1\}^{\lambda}$; $C \xleftarrow{\$} Enc_k(M)$
2. $h = H(C)$; $c_d = h \oplus k$
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 $Recover^{\text{RAONT}}(\boldsymbol{V})$

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. $h = H(C)$; $k = h \oplus k$
6. $M \leftarrow Dec_k(C)$
7. Return M

Figure 6: The dispersal and recovery algorithms defining the robust AONT-RS (RAONT-RS) scheme.

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

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

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

Figure 7: Games for proving Theorem 2, the privacy of the RAONT-RS scheme.

if false shares are submitted, the correct secret can be recovered as long as t correct shares are submitted to $Recover^{RAONT}$. Furthermore, the commitment scheme enables us to learn which servers have been corrupted and thus take any necessary action. However, it is noted that the commitment scheme requires more computation than the use of a canary.

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 [24]. Note that the proof of HK2 is not in the RO model; when proving the security of RAONT-RS, we are forced to prove it in the RO model because the use of the hash function H.

Theorem 2 (Privacy of RAONT-RS) *Let A be an privacy adversary against the RAONT-RS scheme Π_R and let the internal hash function H be a random oracle. Then there is an ind-1 adversary B attacking the indistinguishability of $\mathcal{E}$ such that*

$$\mathbf{Adv}^{Priv}_{\Pi_R}(A) \leq \mathbf{Adv}^{Ind}_{\mathcal{E}}(B) \cdot 4\epsilon(n), \tag{13}$$

where B makes only one query during the deal procedure of Game Ind (as in Figure 2) and the running time of B is that of A plus overhead consisting of one execution of the $Share^{RAONT}$ algorithm of Π_R.

The proof relies on games $G_0 - G_5$, defined in Figure 7. Note that Adversaries G_0 and G_5 are equivalent to those defined in Figure 5 for Theorem 1, the privacy of the AONT-RS scheme. 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. Note this algorithm is not necessarily efficiently implementable.

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

$$\mathbf{Adv}^{Priv}_{\Pi_R}(A) = 2 \cdot \Pr[G^A_0] - 1. \tag{14}$$

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

$$\Pr[G^A_0] = \Pr[G^A_1] = \Pr[G^A_2] + (\Pr[G^A_1] - \Pr[G^A_2]). \tag{15}$$

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

$$\Pr[G^A_1] - \Pr[G^A_2] = \mathbf{Adv}^{Hide}_{\mathcal{CS}}(D_1). \tag{16}$$

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 the commitment scheme. 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 $C \xleftarrow{\$} Enc_k(x_b)$. Adversary 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. This step is not necessarily efficient due to the algorithm DCt. However, D_1 does not have to be computationally bounded.

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, if $b' \neq b$, D_1 submits 0.

Next, we have that

$$\Pr[G^A_2] = \Pr[G^A_3] + \left(\Pr[G^A_2] - \Pr[G^A_3]\right), \tag{17}$$

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 an alternatively generated string k'. We claim that

$$\Pr[G_2^A] = \Pr[G_3^A]. \tag{18}$$

This is justified as the hash function acts as a random oracle. 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 is missing at least λ bits of information about C. Then $c_d = k' \oplus h'$ where $h' \neq h$ is some string unknown to A.

In either case, the adversary learns one of either h or c_d, and learns no information about the other and 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 $\Pr[G_2^A] = \Pr[G_3^A]$.

Next, we have

$$\Pr[G_3^A] = \Pr[G_4^A] + \left(\Pr[G_3^A] - \Pr[G_4^A]\right). \tag{19}$$

We now construct an adversary D_2, also attacking the hiding property of the commitment scheme, such that

$$\Pr[G_3^A] - \Pr[G_4^A] = \mathbf{Adv}^{Hide}_{\mathcal{CS}}(D_2). \tag{20}$$

The construction of D_2 is similar to that of D_1. Adversary D_2 acts as the challenger to A and wishes to use A's advantage to gain an advantage against the hiding property of the commitment scheme $\mathcal{CS}$. Adversary D_2 picks $b \xleftarrow{\$} \{0,1\}$ and runs A. When A submits x_0, x_1 to D_2 during the *Deal* procedure, adversary D_2 generates two values $k, k' \xleftarrow{\$} \{0,1\}^\lambda$ and calculates $C \xleftarrow{\$} Enc_k(x_b)$. Adversary D_2 then computes $H(C) = h$ and $h \oplus k' = c_d$ and 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_2 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, adversary D_2 computes its reply according to the case of the *Corrupt* procedure of games G_3, G_4; that is, adversary D_2 generates a decommittal value $\boldsymbol{R}[i]$ for $\boldsymbol{V}[i]$ and the given $\boldsymbol{H}[i]$, then passes $\boldsymbol{X}[i] \leftarrow \boldsymbol{R}[i]\boldsymbol{V}[i]\boldsymbol{S}_1[i] \ldots \boldsymbol{S}_n[i]$ to A. Again, this step is not necessarily efficiently implementable. When A halts without output b', if $b' = b$, D_2 submits 1 to its challenger, guessing the commitment value $\boldsymbol{H}[i]$ was computed on $\boldsymbol{V}[i]$, rather than $\boldsymbol{C}[i]$. Otherwise, if A halts with output $b' \neq b$, D_2 submits 0, guessing the commitment value $\boldsymbol{H}[i]$ was computed on $\boldsymbol{C}[i]$ rather than $\boldsymbol{V}[i]$.

Game G_5 differs from G_4 only in its *Corrupt* procedure. Clearly

$$\Pr[G_4^A] = \Pr[G_5^A]. \tag{21}$$

Let B be an ind-1 adversary attacking $\mathcal{E}$, as in the proof of Theorem 1. The advantage of B is as described in (11).

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

$$\mathbf{Adv}^{Hide}_{\mathcal{CS}}(D) = \frac{1}{2}\left(\mathbf{Adv}^{Hide}_{\mathcal{CS}}(D_1) + \mathbf{Adv}^{Hide}_{\mathcal{CS}}(D_2)\right). \tag{22}$$

Since Ct is assumed to be $\epsilon(\cdot)-$hiding and D makes at most n queries during the deal procedure, we have

$$\mathbf{Adv}^{Hide}_{\mathcal{CS}}(D) \leq \epsilon(n). \tag{23}$$

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 8: Procedures used by adversary B to respond to queries from A in Theorem 3.

Combining (16), (20), (22) and (23) gives us

$$\begin{aligned} \mathbf{Adv}^{Hide}_{\mathcal{CS}}(D) &\leq \epsilon(n), \\ \frac{1}{2} \cdot \mathbf{Adv}^{Hide}_{\mathcal{CS}}(D_1) + \frac{1}{2} \cdot \mathbf{Adv}^{Hide}_{\mathcal{CS}}(D_2) &\leq \epsilon(n), \\ \big(\Pr[G^A_1] - \Pr[G^A_2]\big) + \big(\Pr[G^A_3] - \Pr[G^A_4]\big) &\leq 2\epsilon(n). \end{aligned}$$

By using $\Pr[G^A_2] = \Pr[G^A_3]$ and $\Pr[G^A_4] = \Pr[G^A_5]$, (as in (18) and (21)) we simplify and rearrange to give us

$$\begin{aligned} \Pr[G^A_1] - \Pr[G^A_5] &\leq 2\epsilon(n) \\ 2\Pr[G^A_1] - 2\Pr[G^A_5] + 1 - 1 &\leq 4\epsilon(n). \end{aligned}$$

Then, we can further rearrange and substitute in the advantage of adversaries A and B, as in (14) and (11), to give

$$\begin{aligned} 2\Pr[G^A_1] - 1 &\leq \big(2\Pr[G^A_5] - 1\big) \cdot 4\epsilon(n) \\ &= \mathbf{Adv}^{Ind}_{\mathcal{E}}(B) \cdot 4\epsilon(n) \\ \mathbf{Adv}^{Priv}_{\Pi_R}(A) &\leq \mathbf{Adv}^{Ind}_{\mathcal{E}}(B) \cdot 4\epsilon(n), \end{aligned}$$

thus completing the proof.

□

4.2 Proof of Robustness

The RAONT-RS scheme can be proven to be recoverable by again adapting the proof by Bellare and Rogaway [22].

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*

$$\mathbf{Adv}^{Rec}_{\Pi_R}(A) \leq n \cdot \mathbf{Adv}^{Bind}_{\mathcal{CS}}(B), \tag{24}$$

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.

Let A be a recoverability adversary against the RAONT-RS scheme Π_R. During *Deal*, A submits x to B. Let $k, C, h, c_d, \boldsymbol{V}, \boldsymbol{H}, \boldsymbol{S_1}, \ldots, \boldsymbol{S}_n, \boldsymbol{X}$ denote the quantities chosen by the RAONT-RS $Share^{RAONT}$ algorithm after x has been submitted. 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

$$\begin{aligned}
\mathbf{Adv}^{Rec}_{\Pi_R}(A) &\leq \Pr[E_3 \cup E_4] && (25)\\
&\leq \Pr[E_1 \cup E_2 \cup E_3 \cup E_4] && (26)\\
&= \Pr[E_1] + \Pr[\overline{E_1} \cap E_2] + \Pr[\overline{E_1} \cap \overline{E_2} \cap E_3]\\
&\quad + \Pr[\overline{E_1} + \overline{E_2} \cap \overline{E_3} \cap E_4]\\
&= \Pr[E_1] + \Pr[\overline{E_1} \cap E_2] + \Pr[\overline{E_2} \cap E_3] + \Pr[\overline{E_2} \cap E_4]. && (27)
\end{aligned}$$

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 [24] 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

$$\Pr[E_1] \leq \sum_{t=1}^{n} \Pr[E_{1,\ell}] = 0. \quad (28)$$

Now we construct adversary B such that

$$\Pr[\overline{E_1} \cup E_2] \leq n \cdot \mathbf{Adv}^{Bind}_{\mathcal{CS}}(B). \quad (29)$$

Adversary B runs A, responding to its *Deal* and *Corrupt* calls via the procedures in Figure 8, 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

$$\Pr[\overline{E_2} \cap E_3] = 0, \quad (30)$$
$$\Pr[\overline{E_2} \cap E_4] = 0. \quad (31)$$

We justify this as follows. If $\boldsymbol{V}'[i] = \boldsymbol{V}[i]$ for all i, then $C = C'$ and $c'_d = c_d$. Therefore (30) holds and so

$$\begin{aligned}
\mathbf{Adv}^{Rec}_{\Pi_R}(A) &= \Pr[E_1] + \Pr[\overline{E_1} \cap E_2]\\
&\quad + \Pr[\overline{E_2} \cap E_3] + \Pr[\overline{E_2} \cap E_4] && (32)\\
&\leq n \cdot \mathbf{Adv}^{Bind}_{\mathcal{CS}}(B), && (33)
\end{aligned}$$

thus completing the proof.

□

5 Comparing RAONT-RS and HK2

In this section, we briefly introduce the HK2 protocol, a robust extension of Krawczyk's SSMS [13] by Bellare and Rogaway [24]. We will then compare HK2 and RAONT-RS.

5.1 The HK2 Scheme

The two algorithms constituting the HK2 protocol, $Share^{\text{HK2}}$ and $Recover^{\text{HK2}}$ are shown in Figure 9.

PROCEDURE $Share^{\text{HK2}}(M)$
1. $k \xleftarrow{\$} \{0,1\}^{\lambda}$
2. $C \xleftarrow{\$} Enc_k(M)$
3. $\boldsymbol{K} \xleftarrow{\$} Share^{PSS}(k)$
4. $\boldsymbol{C} \xleftarrow{\$} Share^{IDA}(C)$
5. For $i \leftarrow 0$ to $n-1$ do
   $(\boldsymbol{H}[i], \boldsymbol{R}[i]) \xleftarrow{\$} Ct(\boldsymbol{K}[i]\boldsymbol{C}[i])$
   $\boldsymbol{S}_i \xleftarrow{\$} Share^{ECC}(\boldsymbol{H}[i])$
6. For $i \leftarrow 0$ to $n-1$ do
   $\boldsymbol{X}[i] \leftarrow \boldsymbol{R}[i]\boldsymbol{K}[i]\boldsymbol{C}[i]\boldsymbol{S}_1[i]\ldots\boldsymbol{S}_n[i]$
7. Return $\boldsymbol{X}$

PROCEDURE $Recover^{\text{HK2}}(\boldsymbol{X})$
1. For $i \leftarrow 0$ to $n-1$ do
   $\boldsymbol{R}[i]\boldsymbol{K}[i]\boldsymbol{C}[i]\boldsymbol{S}_1[i]\ldots\boldsymbol{S}_n[i] \leftarrow \boldsymbol{X}[i]$
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
   $Vf(\boldsymbol{H}[i], \boldsymbol{K}[i]\boldsymbol{C}[i], \boldsymbol{R}[i])) = 0$ then
   $\boldsymbol{K}[i] \leftarrow \Diamond; \boldsymbol{C}[i] \leftarrow \Diamond$
4. $k \leftarrow Recover^{PSS}(\boldsymbol{K})$
5. $C \leftarrow Recover^{IDA}(\boldsymbol{C})$
6. $M \leftarrow Dec_k(C)$
7. Return M

Figure 9: The dispersal and recovery algorithm defining the HK2 scheme.

The HK2 protocol assumes the existence of an encryption system $\mathcal{E}$ with ind-1 security, and an IDA. The IDA used in the HK2 protocol does not have any security requirements-replication could be used and the scheme would remain secure. HK2 also requires a $(t,n)-$PSS and a $(t,n)-$ECC. In order for the scheme to be extended to be robust, a commitment scheme $\mathcal{CS}$ is also used.

Intuitively, the HK2 protocol works as follows. The $Share^{\text{HK2}}$ protocol takes as input a message $M \in \mathcal{M}$. A λ bit key k is randomly generated in Step 1. The plaintext message M is encrypted in Step 2 to give the ciphertext C. The encryption key k is then dispersed via a $(t,n)-$PSS in Step 3 to give an $n-$vector of key shares $\boldsymbol{K}$. The ciphertext C is shared via the $(t,n)-$IDA to return the $n-$vector of ciphertext shares $\boldsymbol{C}$. In Step 4, for each $(\boldsymbol{K}[i]\boldsymbol{C}[i])$, a committal $\boldsymbol{H}[i]$ and decommittal $\boldsymbol{R}[i]$ is calculated and each committal $\boldsymbol{H}[i]$ is shared via a $(t,n)-$ECC. In Step 7, a new vector $\boldsymbol{X}$ is constructed, where each element $\boldsymbol{X}[i]$ consists of the decommittal value $\boldsymbol{R}[i]$, the key share $\boldsymbol{K}[i]$, the ciphertext share $\boldsymbol{C}[i]$ and one share from each of the shared committal values: $\boldsymbol{S}_1[i], \boldsymbol{S}_2[i], \ldots, \boldsymbol{S}_n[i]$. The algorithm returns the vector $\boldsymbol{X}$, where each element $\boldsymbol{X}[i]$ is a share given to player P_i.

The $Recover^{\text{HK2}}$ protocol takes as input shares of the vector $\boldsymbol{X}$, where $\Diamond$ denotes no share submitted. The algorithm then parses each share $\boldsymbol{X}[i]$ into elements $\boldsymbol{R}[i]$, $\boldsymbol{K}[i]$, $\boldsymbol{C}[i]$, $\boldsymbol{S}_1[i]$, $\boldsymbol{S}_2[i]$, $\ldots$, $\boldsymbol{S}_n[i]$ in Step 1. For each i, $\boldsymbol{H}[i]$ is recovered in Step 2. In Step 3, the submitted ciphertext and key shares are verified by the commitment scheme. If the values are successfully verified, the shares of $\boldsymbol{K}$ and $\boldsymbol{C}$ are used to recover k and C in Steps 4 and 5 respectively. Any shares that were not successfully verified are removed from $\boldsymbol{K}$ and $\boldsymbol{C}$ are marked as empty with $\Diamond$. Finally, the algorithm decrypts C in Step 6 and returns the plaintext message M in Step 7.

Note that if the commitment scheme is removed from the protocol, so each player receives just a key share $\boldsymbol{K}[i]$ and a ciphertext share $\boldsymbol{C}[i]$, the scheme is Krawczyks computationally secure scheme, SSMS, as in [13].

Note that any PSS and IDA can be used in HK2 and Krawczyk's SSMS. When Resch and Plank compared the AONT-RS to the SSMS scheme, they decided to use Shamir's PSS and Rabin's IDA as the internal components. There are, however, potentially more efficient internal components that could be used. For example, rather than Shamir's PSS scheme, either Kurihara et al.'s [15] or Chen et al.'s PSS could be used [16]. Rather than Rabin's IDA, the systematic RS-IDA defined by Resch and Plank themselves (described in Section 2.5.1) could be used.

5.2 Comparison

In [22], Resch and Plank briefly compare their AONT-RS scheme to Krawczyk's SSMS scheme [13], and then conduct a thorough performance comparison of the AONT-RS with Rabin's IDA [20] and Shamir's PSS [25].

Rabin's IDA is an IDA as defined in Definition 8, and not a $(t, n)-$secret sharing scheme and thus has no security requirements. As such, Rabin's scheme would not be used to distribute data if there was any requirement for the data to be private. Shamir's PSS is a PSS as defined in Definition 2 and achieves perfect, rather than computational security. In schemes with perfect security, the size of the share given to each player is at least the size of the secret. Shamir's scheme meets this bound and is ideal, but, because of the large share size, would not generally be used to distribute large blocks of data due to this bound.

Krawczyk's SSMS achieves computational security; which is what the AONT-RS scheme aims to achieve. Because of the similar security goals, comparing the SSMS scheme with the AONT-RS appears to be a beneficial comparison. Similarly, comparing the robust AONT-RS with the robust extension of SSMS, which is HK2, appears to be beneficial.

In this section, we will compare the security, share size and efficiency of the RAONT-RS scheme and the HK2 scheme. Note that we do not include the contribution to the share size or complexity made by the commitment scheme $\mathcal{CS}$, or the contribution made to the complexity due to the encryption scheme $\mathcal{E}$. This is because the contribution of both $\mathcal{CS}$ and $\mathcal{E}$ to the RAONT-RS and HK2 are equal. Because of this, we can consider the comparison in this section to also be a comparison of AONT-RS, as defined in Section 3.1 and Krawczyk's SSMS scheme, as in [13].

In order to analyse the schemes better, we assume both schemes use the systematic RS-IDA as in Section 2.5.1 and that HK2 uses an ideal PSS. One suggestion is a PSS by Kurihara et al. [15].

5.2.1 Security.

Both the RAONT-RS and HK2 are proven to be computationally secure secret sharing schemes, but the RAONT-RS is secure in the RO model and the HK2 scheme secure under standard assumptions.

RAONT-RS: We have proven the RAONT-RS to achieve privacy and recoverability, assuming the internal probabilistic encryption scheme $\mathcal{E}$ has ind-1 security, that the hash function H behaves as a RO; that the commitment scheme $\mathcal{CS}$ achieves both the hiding and binding properties and the IDA is equivalent to a $(0, t; n)-$linear ramp scheme. We also

discussed how, if $k, c_d \in \{0,1\}^\lambda$, the ciphertext C should be such that $C \in \{0,1\}^\omega$, where $\omega \geq (t-1)\lambda$.

HK2: The HK2 protocol achieves privacy and recoverability, as proven in [24]. The proof of security is under the assumption that the internal probabilistic encryption scheme $\mathcal{E}$ achieves ind-1 security and that the commitment scheme $\mathcal{CS}$ achieves both the hiding and binding property. One additional assumption is that the $(t,n)-$secret sharing scheme achieves perfect security, as described in Definition 2. The proof for the HK2 protocol is, however, not in the RO model and there are not restrictions on the size of the ciphertext C.

Discussion: Both the RAONT-RS and HK2 achieve computational privacy and recoverability. However, significantly, the security of the RAONT-RS scheme is in the RO model, whereas HK2 is provable in the standard model.

5.2.2 Share Size.

We now compare the size of the shares given to each player in both schemes. Note that we will only consider the share of the data given to each player and not the additional information dealt due to the commitment schemes (the decommittal $\boldsymbol{R}[i]$ and the shares of the committal keys $\boldsymbol{S}_1[i], \ldots, \boldsymbol{S}_n[i]$). This is because the commitment scheme adds the same amount of data to both; the sizes of the shares vary because of the data share and not the additional information making the scheme robust. Therefore this comparison is also applicable to AONT-RS and SSMS.

Assume $M \in \{0,1\}^\omega$ and the encryption scheme $\mathcal{E}$ is length preserving. Encrypt M under some key $k \in \{0,1\}^\lambda$ to calculate the ciphertext $C \in \{0,1\}^\omega$. Assume also that $\omega \geq \lambda(t-1)$ (to prevent attacks against the AONT-RS, as described in Section 3.2).

RAONT-RS: If C and k were distributed via the AONT-RS, the share given to each player would consist of

$$\left\lceil \frac{\omega + \lambda}{t} \right\rceil \tag{34}$$

bits. This is because $(C||c_d)$, a total of $\omega+\lambda$ bits, would be input to the IDA. The IDA would parse $(C||c_d)$ into t elements, each element consisting of $\lceil \frac{\omega+\lambda}{t} \rceil$ bits. These t words would then create a $t-$vector and be multiplied on the left by the $n \times t$ matrix G. Each player would then receive a share from the resulting $n-$vector, which would also have elements of $\lceil \frac{\omega+\lambda}{t} \rceil$ bits.

HK2: If the HK2 scheme were used to distribute C, each share would consist of

$$\left\lceil \frac{\omega}{t} \right\rceil + \lambda \tag{35}$$

bits. This is because each player's share consists of a ciphertext share $\boldsymbol{C}[i]$ and a key share output from the PSS, $\boldsymbol{K}[i]$. The ciphertext share is an output from the IDA, and so would consist of $\lceil \frac{\omega}{t} \rceil$ bits. The key share is output from a PSS which, assuming the PSS is ideal, would consist of λ bits. Thus the total size of the share is the sum of the two parts.

Discussion: We can see that,

$$\left\lceil \frac{\omega + \lambda}{t} \right\rceil \leq \left\lceil \frac{\omega}{t} \right\rceil + \left\lceil \frac{\lambda}{t} \right\rceil \leq \left\lceil \frac{\omega}{t} \right\rceil + \lambda, \tag{36}$$

for all $t \geq 1$. Therefore, the RAONT-RS scheme achieves smaller share sizes than HK2. The share sizes in RAONT-RS are strictly smaller than in HK2 when $t \geq 2$ (which will be

true in general). As t increases, the ratio of the share sizes between RAONT-RS and HK2 will also increase. This is because $\frac{\lambda}{t}$ decreases as t increases, whilst λ is a constant.

Note the majority of each share comes from the size of the ciphertext C, meaning that, although RAONT-RS achieves smaller share sizes, the differences will be minimal if ω is large. If ω is small, however, and λ is comparatively large, RAONT-RS may achieve significantly smaller shares than HK2. This is especially true if t is large.

5.2.3 Efficiency of $Share$.

The number of bitwise XORs required for $Share^{RAONT}$ and $Share^{HK2}$ will now be compared. Note that we have omitted the analysis of both the encryption scheme $\mathcal{E}$ and the commitment scheme $\mathcal{CS}$ This is because the number of XORs required for both $\mathcal{E}$ and $\mathcal{CS}$ will be common to both schemes. Therefore this comparison is also applicable to AONT-RS and SSMS. We will assume the internal IDA used in both schemes is the systematic RS-IDA, as defined in Section 2.5.1.

Assume we wish to distribute some $M \in \{0,1\}^{\omega}$, where $k \xleftarrow{\$} \{0,1\}^{\lambda}$. Assume that $\mathcal{E}$ is a length preserving function, so $C \xleftarrow{\$} Enc_k(M), C \in \{0,1\}^{\omega}$ with $\omega \geq \lambda(t-1)$.

RAONT-RS: After encrypting M, the $Share^{\text{RAONT}}$ algorithm requires one hash function computation and one XOR of λ bits (the values h and k to calculate c_d). Then, $(C||c_d)$ is distributed via the IDA. As multiplication can be implemented via a lookup table, each element of the output $n-$vector require $t-1$ XORs of elements in $GF(2^{\lceil \frac{\omega+\lambda}{t} \rceil})$. This is true for each of the n elements in the vector, thus the IDA requires

$$n(t-1)\left\lceil \frac{\omega+\lambda}{t} \right\rceil = \mathcal{O}(n(\omega+\lambda)) \tag{37}$$

bitwise XORs. Therefore, the total cost of $Share^{\text{RAONT}}$ is one hash function computation and

$$\lambda + n(t-1)\left\lceil \frac{\omega+\lambda}{t} \right\rceil = \mathcal{O}(\lambda(n+1) + n\omega) \tag{38}$$

bitwise XORs. Note that computing one hash is generally very efficient.

HK2: The HK2 protocol, after encrypting M, requires the distribution of the secret k via a PSS. One example of an efficient PSS is a scheme by Kurihara et al.'s scheme. Using Kurihara et al.'s scheme would require $\mathcal{O}(tn\lambda)$ bitwise XORs. After the key is distributed, the ciphertext is distributed via the IDA. As multiplication can be implemented via a lookup table, each element in the output $n-$vector requires $t-1$ XOR of elements, this time, in $GF(2^{\lceil \frac{\omega}{t} \rceil})$. Therefore, the IDA when used in the HK2 scheme requires

$$n(t-1)\left\lceil \frac{\omega}{t} \right\rceil = \mathcal{O}(n\omega) \tag{39}$$

bitwise XORs. Thus, in total, the HK2 protocol requires

$$\mathcal{O}(\lambda(tn) + n\omega) \tag{40}$$

bitwise XORs.

Discussion: The AONT-RS scheme requires fewer bitwise XORs for distribution; this is because $1 + n < tn$ for all $t \geq 2$. Notably, as t increases, the HK2 protocol requires more XORs, whereas the number of XORs for the RAONT scheme is independent of t. This means that, as t increases, the AONT-RS scheme will require fewer bitwise XORs when compared to the HK2 protocol. The RAONT-RS scheme does, however, require the computation of a hash function.

5.2.4 Efficiency of *Recover*.

As before, we discuss only the complexity of the scheme excluding the commitment scheme and decryption. Again, this comparison is also applicable to AONT-RS and SSMS. Assume t players pool their shares in order to recover the data M.

RAONT-RS: In the $Recover^{\text{RAONT}}$ algorithm, the players all have shares of size $\lceil \frac{\omega+\lambda}{t} \rceil$. When t players submit their shares, they would first be input to the $Recover^{IDA}$ algorithm. A $t-$vector of the shares would be constructed and multiplied by the inverted $t \times t$ matrix constructed from G. As in the share procedure, this would require $t-1$ XORs for each element of the output vector. The output vector only has t elements, thus

$$t(t-1)\left\lceil \frac{\omega+\lambda}{t} \right\rceil = \mathcal{O}(n(\omega+\lambda)) \tag{41}$$

bitwise XORs and would recover the ciphertext C and c_d. One hash computation $H(C) = h$ would then be required, followed by one XOR of λ bits, to calculate $k = h \oplus c_d$. The message C would then be decrypted under k. Thus the $Recover^{\text{RAONT}}$ algorithm requires one hash function computation and

$$\mathcal{O}(t(\omega+\lambda)+\lambda) \tag{42}$$

bitwise XORs.

HK2: In the $Recover^{\text{HK2}}$ algorithm, the players all have shares of size $\lceil \frac{\omega}{t} \rceil + \lambda$. After t players submit their shares, the ciphertext share would be separated from the key share. The IDA is used to recover the ciphertext from the ciphertext shares. As before, this requires

$$t(t-1)\left\lceil \frac{\omega}{t} \right\rceil \tag{43}$$

bitwise XORs. The encryption key k would then be recovered via the PSS, which (again, if Kurihara et al.'s PSS is used) would require $\mathcal{O}(tn\lambda)$ bitwise XORs. Thus, the number of bitwise XORs for the HK2 scheme is

$$\mathcal{O}(t\omega + nt\lambda). \tag{44}$$

Discussion: The *Recover* algorithm for the RAONT-RS requires fewer bitwise XOR operations. However, the number of bitwise XORs in the $Recover^{RAONT}$ algorithm is dependent on t, but not on n. The HK2 algorithm is dependent on both t and n.

Again, the RAONT-RS scheme requires one extra operation, the computation of one hash function.

6 Conclusion

In this paper we have generalised the AONT-RS scheme and shown how information is leaked when it is used to distribute ciphertexts shorter than $\lambda(t-1)$, where t is the threshold of the scheme and λ is the security parameter of the encryption scheme. We then proved the AONT-RS scheme achieves computational privacy in the RO model. Next, we extended the scheme to be robust and proved the robust scheme achieves both computational privacy and recoverability in the RO model. Finally, we compared the robust AONT-RS with the HK2 scheme, which is a comparison applicable to the general AONT-RS and Krawczyk's SSMS scheme. We showed the robust AONT-RS scheme achieves weaker security than the

HK2 scheme because its proof is in the RO model, whereas the HK2 scheme is provable under standard assumptions. However, by compromising on security, the AONT-RS scheme achieves smaller share sizes and more efficient dispersal and recovery of data than HK2. This is particularly true when the threshold value t is large.

Further research includes implementing both techniques with a number of different parameters to allow for a more thorough comparison of the performance of the schemes.

References

[1] Philippe Béguin and Antonella Cresti. General short computational secret sharing schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 194–208. Springer, 1995.

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

[3] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, 1993.

[4] George Robert Blakley. Safeguarding cryptographic keys. *Proc. of the National Computer Conference1979*, 48:313–317, 1979.

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

[6] Christian Cachin. On-line secret sharing. In *IMA International Conference on Cryptography and Coding*, pages 190–198. Springer, 1995.

[7] Arun Chandrasekaran, Raj Bala, and Garth Landers. Critical Capabilities for Object Storage. Technical report, March 2016.

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

[9] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2002.

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

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

[12] J. Katz and Y. Lindell. *Introduction to modern cryptography*. CRC Press, 2014.

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

[14] Hugo Krawczyk. Distributed fingerprints and secure information dispersal. In *Proceedings of the twelfth annual ACM symposium on Principles of distributed computing*, pages 207–218. ACM, 1993.

[15] J. Kurihara, S. Kiyomoto, K. Fukushima, and T. Tanaka. A new (k, n)-threshold secret sharing scheme and its extension. In *Information Security*, pages 455–470. Springer, 2008.

[16] Thalia M. Laing, Keith M. Martin, Maura B. Paterson, and Douglas R. Stinson. Localised multisecret sharing. *Cryptography and Communications*, pages 1–17, 2016.

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

[18] Alain Mayer and Moti Yung. Generalized secret sharing and group-key distribution using short keys. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 30–44. IEEE, 1997.

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

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

[21] 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.

[22] 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.

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

[24] 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.

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

[26] Richard Singleton. Maximum distance¡ img src=”/images/tex/62. gif” alt=” q”¿-nary codes. *IEEE Transactions on Information Theory*, 10(2):116–118, 1964.

[27] NIST Computer Security Division's (CSD) Security Technology Group (STG). Block Cipher Modes, Cryptographic Toolkit. NIST. `http://csrc.nist.gov/groups/ST/toolkit/BCM/index.html`, 2013. Accessed: 2016-09-01.

[28] Martin Tompa and Heather Woll. How to share a secret with cheaters. *journal of Cryptology*, 1(3):133–138, 1989.

[29] V Vinod, Arvind Narayanan, K Srinathan, C Pandu Rangan, and Kwangjo Kim. On the power of computational secret sharing. In *International Conference on Cryptology in India*, pages 162–176. Springer, 2003.