Efficient, XOR-based, Ideal $(t, n)-$threshold Schemes

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. We propose a new, lightweight $(t, n)-$threshold secret sharing scheme that can be implemented using only XOR operations. Our scheme is based on an idea extracted from a patent application by Hewlett Packard that utilises error correction codes. Our scheme improves on the patent by requiring fewer randomly generated bits and by reducing the size of shares given to each player, thereby making the scheme ideal. We provide a security proof and efficiency analysis. We compare our scheme to existing schemes in the literature and show that our scheme is more efficient than other schemes, especially when t is large.

Keywords: threshold, secret sharing, perfect, efficient, ideal, error correction

1 Introduction

A $(t, n)-$threshold secret sharing scheme provides a method for distributing a secret k amongst n players in such a way that any t players can uniquely reconstruct the secret and $t-1$ players learn no information about the secret.

Secret sharing can be utilised in distributed systems to store information, such as a decryption key, across n servers. A user wishing to access the encrypted data could retrieve shares from any t servers and combine them to recover the key and decrypt the data. A system such as this would enable greater availability, as there is no single point of failure, and add redundancy, thereby improving reliability in case of failures. Furthermore, secret sharing offers security without the reliance on cryptographic keys. In an encryption system, the decryption key must be stored securely and an adversary that gains access to the key would render the encrypted data insecure. If the key were dispersed via a $(t, n)-$threshold scheme, the adversary would need to extract the shares from t of the n servers in order to recover it. Learning shares from up to $t-1$ servers would reveal no information. A system with these benefits is of particular interest given the growth of constrained devices in potentially insecure locations, such as devices in the Internet of Things. An adversary may find it easy to extract the information stored on a constrained device, but no information would be learnt if the information extracted were a share of a key rather than the key itself.

Blakley and Shamir first introduced $(t, n)-$threshold schemes in 1979 [2,3]. Shamir's scheme achieves minimal share sizes and requires generating minimal randomness. The scheme is elegant but, despite improvements in implementation, has a fairly heavy computational cost because the recovery procedure relies on Lagrange interpolation. Given the increase in constrained devices, there have been efforts to create schemes that achieve the properties of Shamir's scheme but can be implemented using only XOR operations. These schemes claim to be lighter than Shamir's and therefore more applicable to limited devices.

A novel $(t, n)-$threshold scheme that achieves the properties of Shamir's scheme and can be implemented using only XOR operations is presented here. Our scheme is based on a computationally secure scheme by Hewlett Packard (HP) [1], from which we extracted a $(t, n)-$threshold scheme. We have improved their scheme by requiring fewer randomly generated bits and achieving smaller share sizes for each player. We present a proof of security and provide an efficiency analysis. We then compare our scheme with other threshold schemes that have the same properties (minimal share size and XOR based implementation).

Kurihara et al. presented the first XOR based, ideal $(t, n)-$threshold in [5] as a generalisation of the work in [4], which considered $(3, n)-$threshold schemes. Kurihara et al.'s scheme constructs shares by XORing pieces of the secret with multiple random numbers and distributing these amongst the players. Recovery is possible by multiplying a vector consisting of the shares by a matrix generated via Gaussian elimination, which is computationally heavy. Kurihara et al. analysed the efficiency of their scheme and compared it to Shamir's scheme.

Since Kurihara et al.'s scheme, few schemes achieving the same properties have been proposed. Lv et al. proposed one such scheme in [6] and a multisecret analogue in [7], but Wang and Desmedt [8] show that in [6] the size of the shares are smaller than the size of the secret and therefore cannot be correct. Their criticism is, however, only true when more than one secret is being dispersed. Nonetheless, Lv et al.'s scheme does have a number of flaws, such as incompatible matrix-vector multiplications and an incorrect analysis of the number of randomly generated values and will therefore not be further considered.

Wang and Desmedt proposed their own $(t, n)-$threshold scheme that is equivalent to an error correcting code [8]. They proved their scheme to be secure and claimed it could be implemented using only XOR and cyclic shift operations. However, they do not provide an exact distribution algorithm, an efficiency analysis or compare their scheme to the current literature.

1.1 Contributions

Our contribution can be summarised as follows:

- We present a new, ideal $(t, n)-$threshold secret sharing scheme based on an idea from a patent application by HP [1]. The scheme is defined in the Galois Field $GF(2^\ell)$ and requires only XORs and shift operations for both distribution and recovery. The scheme could be generalised to any Galois Field; we have specified $q = 2$ as this is likely to be the case when implemented.

– We provide a proof of security for our scheme and show it is ideal. No proof of security was presented for HP's scheme.
– We analyse the complexity of our proposed scheme with respect to the number of bitwise XOR operations required.
– We compare our scheme to existing XOR based, ideal $(t, n)-$threshold schemes in the literature. In order to conduct this analysis, we have provided the first efficiency analysis of the scheme presented by Wang and Desmedt [8].

1.2 Organisation

This paper is organised as follows. In Section 2 we present notation and provide definitions. In Section 3 we define our scheme and in Section 4 we prove our scheme is an ideal, $(t, n)-$threshold scheme. An efficiency analysis of the distribution and recovery algorithms of the scheme is given in Section 5. In Section 6, we analyse the efficiency of Wang and Desmedt's scheme [8] and compare this with Kurihara et al.'s scheme [5], Shamir's scheme [3] and our scheme.

2 Preliminaries

In this section, the definitions and notation used will be introduced.

2.1 Secret Sharing Schemes

Let $n, t \in \mathbb{N}, t \leq n$. A $(t, n)-$*threshold scheme* is a method of distributing a *secret* k amongst a set of n *players* in such a way that any subset of t players can uniquely recover the secret and any subset of fewer than t players cannot learn any information about the secret.

A $(t, n)-$threshold scheme can be defined using information theoretic notation [9]. Let $\boldsymbol{K}$ 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 share given to the players in the set $A \subseteq \mathcal{P}$.

Definition 1. *A* dealer *distributes a secret k amongst a set $\mathcal{P} = \{P_0, \ldots, P_{n-1}\}$ of n players. Let $t \leq n$ be the threshold and let Γ be a set containing all sets of at least t players. A* $(t, n)-$threshold scheme *consists of two algorithms: Share and Recover. Share is a probabilistic algorithm that takes as input the secret k and outputs an $n-$vector $\boldsymbol{S}$. Each player P_i receives a* share $\boldsymbol{S}[i]$. *Players then input their shares to the Recover algorithm, which satisfies the following properties:*
(1) Any set B of fewer than t players learns no information about the secret: $\forall B \notin \Gamma, H(\boldsymbol{K}|\boldsymbol{B}) = H(\boldsymbol{K})$, and
(2) any set A of at least t players can uniquely recover the secret. Information theoretically: $\forall A \in \Gamma, H(\boldsymbol{K}|\boldsymbol{A}) = 0$.
These two conditions will be referred to as (1) privacy and (2) recoverability.

Note that schemes achieving the privacy requirement are said to have *perfect security*. Shamir proposed the first $(t, n)-$threshold scheme in 1979 [3].

Definition 2 (Shamir's $(t, n)-$threshold scheme). *Let $\mathcal{P}$ be a set of n players $\mathcal{P} = \{P_0, P_1, \ldots, P_{n-1}\}$ and let $p > n$ be some prime. Let $r_1, r_2, \ldots, r_{t-1}$ be $t-1$ values generated uniformly at random from $\mathbb{Z}_p$. For a given secret $k \in \mathbb{Z}_p$, let $f \in \mathbb{Z}_p[x]$ be the polynomial defined by*

$$f(x) = k + r_1 x + r_2 x^2 + \cdots + r_{t-1} x^{t-1}.$$

Allocate to each player P_i the share $\boldsymbol{S}[i] = f(i+1)$.

Any set of t or more players can use polynomial interpolation on their shares to recover the polynomial $f(x)$ and thus calculate $k = f(0)$. However, given fewer than t shares there exists a polynomial in $\mathbb{Z}_p[x]$ of degree $t-1$ for any $k \in \mathbb{Z}_p$. Thus $t-1$ shares reveals no information about k.

In any $(t, n)-$threshold scheme distributing a key of λ bits requires the generation of a minimum of $\lambda(t-1)$ bits of randomness for distribution [19].

The *information rate* ρ of a scheme is the ratio of the size of the secret to the size of the largest share. In any $(t, n)-$threshold scheme the share given to each player must be at least the size of the secret [9], so $\rho \geq 1$. Schemes that meet this bound have every share equal to the size of the secret and are called *ideal*. Shamir's scheme is an ideal scheme.

Schemes with smaller share sizes can be constructed by relaxing the requirement for the scheme to achieve perfect security, meaning that sets of fewer than t players learn some information about the secret. Ramp schemes achieve this.

Definition 3. *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. A set of t_0 or fewer players reveals no information about the secret.*

Observe that a $(t, n)-$threshold scheme is a $(t-1, t; n)-$ramp scheme. There is no bound on the amount of information a set of between t_0 and t_1 players can learn about k. In a $(0, t, n)-$threshold scheme, there are no security constraints; only recoverability for t players must be satisfied. In Section 2.3 it is shown that a $(0, t; n)-$ramp scheme is equivalent to an information dispersal algorithm.

If we wish the security of the ramp scheme to be maximised with respect to the limit on the size of each share then the information theoretic knowledge about the secret is likely to increase linearly with respect to the number of participants pooling their shares. This motivates the following definition [10].

Definition 4. *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 $t_0 \leq r \leq t_1$,*

$$H(\boldsymbol{K}|\boldsymbol{A}) = \frac{t_1 - r}{t_1 - t_0} H(\boldsymbol{K}). \tag{1}$$

Intuitively, a linear $(t_0, t_1; n)-$ramp scheme reveals no information about the secret k to a set containing up to t_0 players. Then, for every further player contributing a share, a fixed amount of information is learnt about k. This continues in a linear fashion until t_1 players have pooled their shares and k 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 k.

2.2 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 [11].

Definition 5. *An* error correcting code *(ECC)* C *of* length n *over a finite alphabet* F *is a subset of* F^n*. The elements of* C *are called* codewords*. The* size *of* C *is* $|C| = m$*. The* minimum distance *of* C *is the minimum Hamming distance between any two distinct codewords in* C *and is denoted by* d*.*

ECCs are able to detect and correct a number of errors. A code C is e_1-*error detecting* if, whenever a codeword $u \in C$ is sent and between 1 and e_1 errors occur, the received word v is not a codeword, so the receiver will know something has gone wrong in the channel and the error will be detected. A code C is e_2-*error correcting* if, whenever $u \in C$ is a codeword, and v is a word of length n over F such that $v \neq u$ and the Hamming distance between u and v is at most e_2, then v is decoded to u using nearest neighbour decoding.

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

One important type of ECC is a maximum distance separable code [11].

Definition 6. *A* maximum distance separable *(MDS) code is a linear code that meets the Singleton bound:* $d = n - t + 1$*.*

MDS codes have the maximum possible Hamming distance between codewords and each can be separated into message symbols and check symbols. A code in which the message string appears in the codeword is called *systematic*. In an MDS code, recovery of a codeword is possible from any t of the n symbols. MDS codes are e error correcting and $2e = n - t$ error detecting. For a received word with a errors and b erasures, the codeword will be uniquely recovered if $2a + b < 2e$. One example of an MDS code is a Reed Solomon (RS) code [12].

The notions of $(t, n)-$threshold schemes and MDS codes are closely related. In 1983, Karnin et al. observed that every ideal $(t, n)-$threshold scheme determines a unique MDS code and vice versa [13]. Furthermore, a $[n, t, n-t+1]-$MDS code is equivalent to a linear $(0, t; n)-$ramp scheme [14]. An RS code is a type of MDS code, and thus is also equivalent to a linear ramp scheme.

A $(t, n)-$*optimal erasure code* is an ECC that transforms a message of t symbols into a codeword of length n such that any t received symbols from the codeword allow reconstruction of the message. An erasure code cannot necessarily correct errors, but can correct up to $n-t$ erasures [15]. A $(t, n)-$erasure code is equivalent to a $(t, n)-$information dispersal algorithm.

2.3 Information Dispersal Algorithms

Information dispersal was first introduced by Rabin in 1989 [16].

Definition 7. *Let $t, n \in \mathbb{N}, t \leq n$. A $(t,n)-$information dispersal algorithm (IDA) consists of a message space $\mathcal{M}$ and two algorithms Share and Recover. Share takes as input a message $m \in \mathcal{M}$ and outputs an $n-$vector $\boldsymbol{S}$. Recover takes as input elements of the vector $\boldsymbol{S}$. If at least t elements are submitted correctly to Recover, 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 $(t,n)-$threshold scheme, but there are no privacy requirements on an IDA. Any $(t,n)-$threshold scheme is an IDA, but it is possible to achieve smaller share sizes in a $(t,n)-$IDA than in a $(t,n)-$threshold scheme.

Systematic IDA Using systematic erasure codes, Plank and Resch [17] present a systematic $(t,n)-$IDA consisting of two algorithms: $Share^{\text{IDA}}$ and $Recover^{\text{IDA}}$.

$Share^{\text{IDA}}$ is a probabilistic algorithm that takes as input a message M to be distributed between n players. M is parsed into a $t-$vector $\boldsymbol{M}$ where each element is in the Galois Field $F = GF(2^{\lambda})$. Let G be a publically known $n \times t$ binary matrix such that the first t rows form the $t \times t$ identity matrix I_t and the final $n - t$ rows are filled with bits such that any t of the n rows of G are linearly independent. Multiply G and $\boldsymbol{M}$ to calculate an $n-$vector $\boldsymbol{C}$, 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 $1 \times d = d \in F$.

Each player P_i is then given the element $\boldsymbol{C}[i]$ as their share. Note that, because the first t rows of G form the identity matrix, the first t elements of $\boldsymbol{C}$ will be identical to elements in $\boldsymbol{M}$. This matrix-multiplication is shown in (2).

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ G[t][0] & G[t][1] & \dots & G[t][t-1] \\ \vdots & \vdots & \ddots & \vdots \\ G[n-1][0] & G[n-1][1] & \dots & G[n-1][t-1] \end{pmatrix} \begin{pmatrix} \boldsymbol{M}[0] \\ \boldsymbol{M}[1] \\ \dots \\ \boldsymbol{M}[t-1] \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}[0] \\ \boldsymbol{M}[1] \\ \dots \\ \boldsymbol{M}[t-1] \\ \boldsymbol{C}[t] \\ \dots \\ \boldsymbol{C}[n-1] \end{pmatrix} \quad (2)$$

In order to recover M, t shares are submitted to $Recover^{\text{IDA}}$. A new $t-$vector $\boldsymbol{C}'$ is created consisting of the t pooled shares. A $t \times t$ matrix G' is then formed, which consists of the t rows of G corresponding to the shares pooled. This matrix is then inverted and multiplied by the vector $\boldsymbol{C}'$ to return $\boldsymbol{M} = (G')^{-1} \cdot \boldsymbol{C}'$.

Plank and Resch's systematic IDA is a systematic variant of Rabin's IDA [16], which uses a matrix G consisting entirely of random bits. The systematic

version is more efficient because only the final $n-t$ elements of $\boldsymbol{C}$ are encoded; the first t elements can be directly copied from the vector $\boldsymbol{M}$.

As mentioned in Section 2.1, an IDA is equivalent to a $(0,t;n)-$ramp scheme. In particular, Plank and Resch's systematic IDA is a linear $(0,t;n)-$ramp scheme if $M \in \{0,1\}^\ell$ has *maximal entropy*, meaning that $H(\boldsymbol{M}) = 2^\ell$.

Theorem 1. *Plank and Resch's systematic IDA is a linear $(0,t;n)-$ramp scheme if the message $M \in \{0,1\}^\ell$ has maximal entropy $H(M) = 2^\ell$.*

Proof. A set of t players are able to recover M by constructing the vector $\boldsymbol{C}'$, creating the corresponding G' and then calculating $(G')^{-1} \cdot \boldsymbol{C}'$. Therefore the systematic IDA satisfies the recoverability property of a ramp scheme.

Now we must show that the IDA is linear. Recall from Section 2.2 that a $(t_0,t_1;n)-$ramp scheme is linear if (1) is satisfied. So, we must show that, for any set of players A, such that $|A| = r$ for $0 \leq r \leq t$,

$$H(M|\boldsymbol{A}) = \frac{t-r}{t}H(M).$$

As $H(M) = 2^\ell$, when M is parsed into t equal sized elements to form the $t-$vector $\boldsymbol{M}$, each element has entropy $\boldsymbol{M}[i] = \frac{2^\ell}{t}$. So each element of $\boldsymbol{M}$ learnt reduces the entropy of M by exactly $\frac{2^\ell}{t}$. Thus,

$$H(M|\boldsymbol{A}) = 2^\ell - \left(r\frac{2^\ell}{t}\right) = \frac{2^\ell(t-r)}{t} = \frac{t-r}{t} \times 2^\ell = \frac{t-r}{t}H(M),$$

as required. Therefore the systematic IDA is a linear $(0,t;n)-$ramp scheme. □

3 An Efficient $(t,n)-$threshold Scheme

In this section, we present our $(t,n)-$threshold scheme, based on an idea extracted from HP's scheme [1]. Our scheme improves HP's by requiring the generation of fewer random bits, by decreasing the size of each player's share thereby making the scheme ideal, and by using the systematic IDA described in Section 2.3 to minimise encoding. Our scheme is defined in the Galois Field $F = GF(2^\ell)$, as this is likely the chosen field for implementation. The scheme could, however, be generalised to any Galois field $GF(q^\ell)$.

3.1 Share

Our scheme constitutes two algorithms, *Share* and *Recover*. Both algorithms are presented in Figures 1 and 2 respectively.

Share(k) is a probabilistic algorithm that takes as input a secret $k \in \{0,1\}^\lambda$. The secret k is considered as a string of t words, with each word consisting of $\lceil \frac{\lambda}{t} \rceil$ bits, by parsing k into t elements. Let $GF(2^{\lceil \frac{\lambda}{t} \rceil})$, then $k \in F^t$. If λ is divisible by t, k will parse exactly into t words. If not, k must be padded with exactly

$(-\lambda) \mod t$ elements to ensure each word is an element in F and $k \in F^t$. For ease of notation, it will be assumed that λ is divisible by t throughout.

Step 1 of the algorithm randomly generates $t-1$ *dummy keys*, labelled $r_1, \ldots, r_{t-1} \in GF(2^\lambda)$. As was done with k, these are parsed into t words, so each $r_i \in F^t$. In Step 2, k and the dummy keys are XORed to produce $k' \in F^t$.

All the dummy keys $r_1, \ldots, r_{t-1}$ and k' are then treated as $t-$vectors over F and independently dispersed via the $Share^{\mathrm{IDA}}$ algorithm described in Section 2.3. This results in t vectors of length n: $\boldsymbol{R_1}, \ldots, \boldsymbol{R_{t-1}}$ and $\boldsymbol{K'} \in F^n$. The first t elements of the output from the IDA are equal to the t elements from the input vector. The final $n-t$ elements are check symbols.

Elements of each of these $n-$vectors are then given to the n players in Step 5 in such a way that every player receives t elements: each element will be from a distinct vector and will be from a different position in each vector. Let $\boldsymbol{R_i}[j]$ denote the j^{th} element in the vector $\boldsymbol{R_i}$. One possible distribution process can be illustrated by constructing a $t \times n$ matrix M, where each $n-$vector output by the systematic IDA defines a row of M, where

$$M = \begin{pmatrix} \boldsymbol{K'}[0] & \boldsymbol{K'}[1] & \ldots & \boldsymbol{K'}[n-2] & \boldsymbol{K'}[n-1] \\ \boldsymbol{R_1}[0] & \boldsymbol{R_1}[1] & \ldots & \boldsymbol{R_1}[n-2] & \boldsymbol{R_1}[n-1] \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \boldsymbol{R_{t-1}}[0] & \boldsymbol{R_{t-1}}[1] & \ldots & \boldsymbol{R_{t-1}}[n-2] & \boldsymbol{R_{t-1}}[n-1] \end{pmatrix}. \tag{3}$$

A new matrix can then be constructed from M by shifting elements in row i, for $0 \leq i \leq t$, i places to the left, resulting in the matrix

$$M' = \begin{pmatrix} \boldsymbol{K'}[0] & \boldsymbol{K'}[1] & \ldots & \boldsymbol{K'}[n-2] & \boldsymbol{K'}[n-1] \\ \boldsymbol{R_1}[1] & \boldsymbol{R_1}[2] & \ldots & \boldsymbol{R_1}[n-1] & \boldsymbol{R_1}[0] \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \boldsymbol{R_{t-1}}[t] & \boldsymbol{R_{t-1}}[t+1] & \ldots & \boldsymbol{R_{t-1}}[t-2] & \boldsymbol{R_{t-1}}[t-1] \end{pmatrix}. \tag{4}$$

The elements in column i of M' are then concatenated and given to P_i as their share. Label the share given to player P_i for $0 \leq i \leq n-1$ as $\boldsymbol{S}[i] \in F^t$.

3.2 Recover

The $Recover(\boldsymbol{S})$ algorithm requires the input of shares from at least t players. Steps 1, 2 and 3 check that a sufficient number of shares are contributed. If $\boldsymbol{S}[i] = \Diamond$, player i has not contributed their share to the *Recover* algorithm. If fewer than t shares are submitted, the algorithm will fail and output $\perp$. Otherwise, Step 4 parses each player's share $\boldsymbol{S}[i]$ into its t elements and, in Steps 5 and 6, $t-$vectors are constructed from the available shares. These are used to recover k' and $r_1, r_2, \ldots, r_{t-1}$ via the $Recover^{\mathrm{IDA}}$ algorithm, as in Section 2.3. These recovered values are then XORed in Step 7 to retrieve the key, k.

In our scheme, $t-1$ dummy keys are generated. This is an improvement on HP's scheme where t are generated [1]. The generation of fewer dummy keys results in smaller dimensions for the matrices M and M', thereby decreasing the size of the shares given to each player. Therefore, in our scheme, each player

Procedure $Share(k)$
1. For $i \leftarrow 1$ to $t-1$ do
   $r_i \xleftarrow{\$} \{0,1\}^\lambda$
2. $k' = k \oplus r_1 \oplus \cdots \oplus r_{t-1}$
3. $\boldsymbol{K'} \leftarrow Share^{\mathrm{IDA}}(k')$
4. For $i \leftarrow 1$ to $t-1$ do
   $\boldsymbol{R_i} \leftarrow Share^{\mathrm{IDA}}(r_i)$
5. For $i \leftarrow 0$ to $n-1$ do
   $\boldsymbol{S[i]} \leftarrow \boldsymbol{K'}[(i)]\boldsymbol{R_1}[i+1 \mod n]$
   $\ldots \boldsymbol{R_j}[i+j \mod n]$
   $\ldots \boldsymbol{R_{t-1}}[i+(t-1) \mod n]$
6. Return $\boldsymbol{S}$

Fig. 1: Our $(t,n)-$threshold $Share$ algorithm

Procedure $Recover(\boldsymbol{S})$
1. $j = 0$
2. For $i \leftarrow 0$ to $n-1$ do
   If $\boldsymbol{S}[i] \neq \Diamond$, then $j = j+1$
3. If $j < t$, return $\perp$
4. For $i \leftarrow 0$ to $n-1$ do
   $\boldsymbol{K'}[(i)]\boldsymbol{R_1}[i+1 \mod n]$
   $\ldots \boldsymbol{R_j}[i+j \mod n] \ldots$
   $\boldsymbol{R_{t-1}}[i+(t-1) \mod n] \leftarrow \boldsymbol{S[i]}$
5. $k' \leftarrow Recover^{\mathrm{IDA}}(\boldsymbol{K'})$
6. For $i \leftarrow 1$ to $t-1$ do
   $r_i \leftarrow Recover^{\mathrm{IDA}}(\boldsymbol{R_i})$
7. $k = k' \oplus r_1 \oplus r_2 \oplus \cdots \oplus r_{t-1}$
8. Return k

Fig. 2: Our $(t,n)-$threshold $Recover$ algorithm

receives t elements in F, rather than $t+1$ elements. Finally, we specify a systematic IDA, rather than using a general RS code, to distribute k' and the dummy keys. The systematic IDA requires the encoding of only the final $n-t$ words, thereby making the scheme more efficient.

It will be shown in the next section that generating only $t-1$ dummy keys and using a systematic IDA is sufficient to ensure the $(t,n)-$threshold scheme achieves the recoverability and privacy properties required, as in Definition 1.

4 Security Analysis

It will now be proved that the $(t,n)-$threshold scheme presented in Section 3 satisfies the recoverability and privacy requirements, as in Definition 1. The original HP scheme has no such analysis [1]. The proof given here can easily be adapted to prove the security of the key distribution in the HP scheme.

The structure of the proof is as follows. In Lemma 1, it is shown that any distribution of elements from the matrix M to the n players that allows any t players to learn at least t shares in every row of M will allow recovery of the secret. Theorem 2 then shows that the distribution of elements from M via M' satisfies this condition, thus the scheme achieves recoverability. Lemma 2 shows that any distribution of elements from the matrix M to the players that allows no more than $t-1$ players to learn at most $t-1$ elements in each column of M achieves privacy. Theorem 3 shows that the distribution of elements from M via M' satisfies this property, thus achieving privacy. Together, Theorems 2 and 3 show that our scheme satisfies the requirements to be a $(t,n)-$threshold scheme, as in Definition 1. It will then be shown that our scheme is ideal.

4.1 Recoverability

Lemma 1. *Let n players be allocated elements from the matrix M. If any set of at least t players learns at least t elements in every row of M, k can be recovered.*

Proof. Assume fragments from the $t \times n$ matrix M are allocated such that any t players learn at least t shares in every row. We wish to show that a set of (at least) t players can pool their shares and learn $k', r_1, \ldots, r_{t-1}$, then recover k. Each row $M[i]$ of M, when transposed and considered as an $n-$vector, is the output of the systematic $(t, n)-$IDA. In particular, $M[0]^T \leftarrow Share^{\mathrm{IDA}}(k')$ and $M[i]^T \leftarrow Share^{\mathrm{IDA}}(r_i)$ for $1 \leq i \leq t-1$. For any row $M[i]$ of M, t players can form a $t-$vector $M'[i]$ and use this vector as input to $Recover^{\mathrm{IDA}}$. This will return the dispersed value: k' if $i = 0$, or r_i if $1 \leq i \leq t-1$. The players can repeat this procedure for every row of M and reconstruct all the values $k', r_1, \ldots, r_{t-1}$. Once these values are obtained, they are able to add them and output k. □

Theorem 2. *The $(t, n)-$threshold scheme satisfies the property that any set of t players learn at least t elements in every row of M.*

Proof. Each player is given a column of the matrix M', where M' is formed by shifting row i of M, for $0 \leq i \leq t-1$, i places to the left. As each player is allocated a distinct column of M', each player is necessarily given a distinct element from every row. Therefore, when any t players pool their shares, there will be t distinct elements in every row. □

Theorem 2 shows that the scheme meets the requirements in Lemma 1 and hence meets the recoverability requirements of a $(t, n)-$threshold scheme.

4.2 Privacy

Now we must prove that the privacy requirement is also satisfied. Intuitively, the following lemma shows that an unauthorised set of players must be prevented from learning all elements in a given column of M, otherwise the players could calculate the corresponding part of k by XORing all elements in that column.

Lemma 2. *If elements from M are allocated such that any set A of at most $t-1$ players learn no information about at least one element from every column, then no information is learnt about k. In information theoretic terms, $H(\boldsymbol{K}) = H(\boldsymbol{K}|S)$, where S is the set of shares given to the players in A.*

Proof. Let k be a string of λ bits. Assume a set A of at most $t-1$ players collectively knows a set S of elements from the matrix M. Assume that players in A can learn no information about at least one element from every column in M. We must first note that the IDA used in the $(t, n)-$threshold scheme is systematic, therefore the result of XORing any of the first t columns of the matrix M will give the corresponding fragment of the key k. Similarly, if we add any of the final $n-t$ columns of M, the output will be the corresponding entry of the codeword vector for $\boldsymbol{k} \Rightarrow Share^{\mathrm{IDA}}(K)$.

For any given column j, for $0 \leq j \leq n-1$, S can contain any number of elements (but not every element) of $M[j]$, the j^{th} column of M. So S will contain at most $t-1$ of the t elements in each column. Without loss of generality, choose a column j; we will prove that no information is learnt about k from this individual column. This argument can then be applied to every other column of M.

Denote the set of elements in column j and not in S as S'_j. Note that $|S'_j| \geq 1$. Let s_j be the XOR of all the elements in S. Let b_j be the XOR of all elements in S'_j, so b_j is the XOR of all elements in column j that are unknown to A.

Assume the key k is also distributed via the IDA, resulting in the vector $\boldsymbol{K}$. Denote the j^{th} element of $\boldsymbol{K}$ as $\boldsymbol{K}[j]$. Note that $s_j \oplus b_j = \boldsymbol{K}[j]$. As the dummy keys are all randomly generated, each value r_i has entropy 2^λ. Thus each element of $\boldsymbol{R}_i$ has entropy $\frac{2^\lambda}{t}$, and so $H(b_j) = \frac{2^\lambda}{t}$.

Now, we can equate this to a one-time pad. The value s_j is equivalent to a known ciphertext and b_j is equivalent to a key. The XOR of these would reveal the plaintext message $\boldsymbol{K}[j]$. As b_j has entropy $H(b_j) = 2^\lambda$, the value $\boldsymbol{K}[j]$ also has entropy 2^λ. Thus no information is learnt about $\boldsymbol{K}[j]$. □

Theorem 3. *The $(t, n)-$threshold scheme satisfies the condition that any set of $t-1$ players learns no information about at least one element in every column.*

Proof. In the scheme, each player is given a column of the matrix M', so each player will receive exactly one element from each row of M. Therefore any set of $t-1$ players will learn exactly $t-1$ elements of each row. As the IDA is a linear $(0, t; n)-$ramp scheme and because each of the dummy keys and k are generated uniformly at random, players with only $t-1$ elements are unable to learn any further elements from each row. Each player will also be given elements that come from $n-t$ distinct columns of M. Therefore a set of up to $t-1$ players can pool their shares and learn at most $t-1$ shares in each column. □

Theorem 3 proves that the scheme meets the requirements of Lemma 2 and thus satisfies the privacy requirement. Therefore the scheme presented in Section 3 satisfies both the recoverability and privacy requirements to be a $(t, n)-$threshold scheme. Note that this security analysis is not specific to the systematic IDA and any general $(0, t; n)-$ramp scheme could be used.

4.3 Information Rate

Finally, we will comment on the information rate of the scheme, which is an improvement of HP's original scheme [1] by a factor of $\frac{t}{t+1}$.

Theorem 4. *Let $k \in \{0,1\}^\lambda$. The $(t, n)-$threshold scheme has an information rate of 1 if λ is divisible by t and is therefore ideal. If λ is not divisible by t, each share has $\lambda \mod t$ more elements than the secret.*

Proof. In general, $(-\lambda) \mod t$ bits of padding will be added to k, so that each element in the matrix M will consist of $\left\lceil \frac{\lambda}{t} \right\rceil$ bits. Each player will receive t elements from M, and thus the size of each player's share will be $\left\lceil \frac{\lambda}{t} \right\rceil t$ bits. Therefore, the information rate is calculated as

$$\rho = \lambda \div \left(\left\lceil \frac{\lambda}{t} \right\rceil t \right).$$

If λ is divisible by t, then $\rho = 1$ and the scheme is ideal. □

5 Efficiency Analysis

Our $(t, n)-$threshold scheme can be implemented using only XORs and cyclic shifts. In this section, the complexity of both *Share* and *Recover* with respect to the number of bitwise XORs is computed. We also consider what computations can be pre-computed, before the dealer has knowledge of the key k.

5.1 Complexity of *Share*

The *Share* protocol requires the generation of $t-1$ random strings (dummy keys r_i for $1 \leq i \leq t-1$) of λ bits. These dummy keys are XORed with the secret k to output k'. In total, this requires $t-1$ XORs of λ bit strings. However, $t-2$ of these XORs can be pre-computed (XORing the dummy keys).

The dummy keys and k' are then dispersed via the systematic IDA. This is computationally the most expensive operation. Each value is treated as a $t-$vector (where element in $F = \{0,1\}^{\frac{\lambda}{t}}$) and multiplied by an $n \times t$ binary matrix G to output an $n-$vector. As the first t rows of the matrix G form the $t \times t$ identity matrix, the first $t-$elements of the $n-$vector will be identical as those in the input $t-$vector. Therefore, only the final $n-t$ elements of the $n-$vector need to be computed, which is done by multiplying the final $n-t$ rows of G with the input $t-$vector. This computation requires $(n-t)(t-1)$ XORs of $\frac{\lambda}{t}$ bit strings. No multiplications are required as G is a binary matrix and so multiplication can be implemented as a lookup table. Each of the $t-1$ dummy keys $r_i, 1 \leq i \leq t-1$ and k' must be dispersed via the IDA, resulting in $t(n-t)(t-1)$ XORs of $\frac{\lambda}{t}$ bit strings.

Of these XORs, the distribution of the random values via the systematic IDA can be pre-computed. There are $t-1$ random values to be dispersed, meaning that $\mathcal{O}(nt) \cdot \lambda$ bitwise XORs can be pre-computed. The dispersal of k' cannot be pre-computed, which requires $\mathcal{O}(n) \cdot \lambda$ bitwise XORs.

Therefore, without pre-computation, a total of

$$\frac{t(n-t)(t-1)\lambda}{t} + (t-1) \cdot \lambda = \mathcal{O}(nt) \cdot \lambda$$

bitwise XORs are required. Of these, $\mathcal{O}(tn) \cdot \lambda$ can be pre-computed and $\mathcal{O}(n) \cdot \lambda$ require knowledge of k.

We do not consider the computation costs of constructing the matrix G, as it is easy to construct and is a public matrix that can be pre-computed and reused.

5.2 Complexity of *Recover*

Recovering the dummy keys and k' is achieved by multiplying the t contributed shares with a $t \times t$ matrix M constructed from the relevant rows of G. The complexity of the *Recover* algorithm can vary dependent on which rows of M are used. The best case is that all t shares submitted to *Recover* will correspond to the t rows of M that form the identity matrix I_t. In this case, no XORs will

be required to recover the dummy keys and k'. Therefore recovery requires only $t-1$ XORs of strings of length λ to calculate k, totalling $\mathcal{O}(t) \cdot \lambda$ bitwise XORs.

In the worst case scenario, the $n-t$ corresponding rows of M that are not a part of the identity matrix I_t will be used to reconstruct the dummy keys and k'. This would require $\mathcal{O}(n-t)(t-1)$ XORs of $\frac{\lambda}{t}$ bit strings. This must be repeated for each of the t values, totalling $\mathcal{O}(nt) \cdot \lambda$ bitwise XORs.

If each player contributes a share to *Recover* with equal probability, the best-case scenario happens with probability $1 \div \binom{n}{t}$, as exactly one $t-$set of players out of a possible $\binom{n}{t}$ sets will yield the best case. If players do not contribute shares with equal probability, it may be possible to give players more likely to contribute a share that corresponds to a row of the identity matrix to minimise computations. If $t = n$, the best case scenario will always occur.

6 Comparison to Other Schemes

In this section, we discuss the efficiency of other $(t, n)-$threshold schemes, then compare the current schemes to our proposed scheme.

6.1 Other Schemes

Kurihara et al. [5] present a complexity analysis on the number of bitwise XORs required in their scheme. For some prime $p \geq n$, their scheme requires the generation of $(t-1)(p-1)\lceil \frac{\lambda}{p-1} \rceil$ bits. Their *Share* algorithm requires $\mathcal{O}(nt) \cdot \lambda$ bitwise XORs. Of these XORs, $\mathcal{O}(nt) \cdot \lambda$ require knowledge of only the random strings and can be pre-computed, whereas $\mathcal{O}(n) \cdot \lambda$ cannot. Each player's share is $\lceil \frac{\lambda}{p-1} \rceil (p-1)$ bits. If λ is divisible by $p-1$, the scheme is ideal.

To recover the secret, a matrix M is computed using Gaussian elimination, which has complexity $\mathcal{O}(t^3 p^3)$. As M can be pre-computed and reused, we include the computation of M in Figure 3 as optional. Recovery then requires $\mathcal{O}(tp) \cdot \lambda$ bitwise XORs. As p is close to n, we let $p = n$ in Figure 3.

Wang and Desmedt [8] present an efficient, ideal $(t, n)-$threshold scheme for $\lambda \geq n$, equivalent to an $[q, t, q-t+1]-$MDS code, for some prime $q \geq \lambda + 1$.

A specific distribution algorithm is not defined. Instead, they define the scheme to be a collection of $(q-1) \times q$ matrices that satisfy $q(q-t)$ linear constraints, with each column of the matrix forming a share for a player, excluding the first which defines the secret. Two ways of calculating the matrices are suggested and either requires only XORs and bit shifts.

One of their suggestions is as follows. Choose a prime $q \geq \lambda + 1$. Randomly generate t elements $f_0, \ldots, f_{t-1} \in \{0,1\}^q$ and create a $q \times t$ matrix F of these values, where each column corresponds to a random string. A $t \times n$ binary matrix G is then constructed where elements are coefficients of the generator polynomial of the Galois Field $GF(2^\lambda)$. Generating the elements of G is done by choosing a primitive element α of $GF(2^\lambda)$ and calculating

$$g(x) = (x-1)(x-\alpha)\ldots(x-\alpha^{n-t-1}) = g_0 + g_1 x + \cdots + g_{n-t} x^{n-t}.$$

This requires $(n-t)^2-1$ XORs, while multiplication can be implemented via a lookup table, as before. So the pre-computation required is $\mathcal{O}(n^2)$ bitwise XORs.

The two matrices F and G are then multiplied together. As G is a binary matrix, multiplication can be implemented as a lookup table and so only $q(t-1)n$ bitwise XORs are required. As q is near to λ, we note this as $\mathcal{O}(nt)\cdot\lambda$ bitwise XORs in Figure 3. Each player is given a column from the output matrix $n-$vector, which is a string of q bits. If $q=\lambda+1$, the scheme is ideal.

The recovery procedure for the scheme is equivalent to a decoding procedure presented by Blaum and Roth [18] that decodes an array code with at most r erasures and no errors. The procedure requires $\mathcal{O}(r(q^2+r))$ XOR operations. The number of erasures Wang and Desmedt's scheme can correct is $n-t$, thus the decoding algorithm requires $\mathcal{O}((n-t)(q^2+n-n))=\mathcal{O}(nq^2)$ XORs. The elements in F each have λ bits, and thus there are $\mathcal{O}(nq^2)\cdot\lambda$ bitwise XOR operations. The value q is of the same magnitude as λ, and so the recovery algorithm requires $\mathcal{O}(nq^2)\cdot\lambda=\mathcal{O}(n)\cdot\lambda^3$ bitwise XOR operations.

6.2 Discussion

Figure 3 presents a summary of the number of bitwise XORs required in our scheme, the two schemes considered and Shamir's scheme. The second and third columns consider the complexities of the distribution algorithms if as much pre-computation is done as possible. The fourth column considers the distribution complexities if no pre-computation is possible and the fifth considers recovery.

The table does not consider the share size and the randomness required for each scheme. This is because all schemes are essentially ideal and all schemes (apart from Wang and Desmedt's) require the generation of a minimum amount of randomness ($(t-1)\lambda$ bits [19]). Wang and Desmedt's scheme requires the random generation of $t\times m$ bits, where $m\geq\lambda+1$.

Note that much of the computation for the distribution in Shamir's scheme can be pre-computed. This is because the dealer possesses all coefficients of the polynomial f and most of the inputs to f are fixed. Therefore sharing the secret would require only adding the secret to each pre-computed value.

Scheme	Pre-computation	*Share* complexity	*Share* without pre-computation	*Recover* complexity
Our Scheme	$\mathcal{O}(nt)\cdot\lambda$	$\mathcal{O}(n)\cdot\lambda$	$\mathcal{O}(nt)\cdot\lambda$	$\mathcal{O}(t)\cdot\lambda\leq x\leq\mathcal{O}(nt)\cdot\lambda$
Kurihara et al. [5]	$\mathcal{O}(n^3t^3)$ or $\mathcal{O}(nt)\cdot\lambda$	$\mathcal{O}(n)\cdot\lambda$	$\mathcal{O}(n^3t^3)$ or $\mathcal{O}(nt)\cdot\lambda$	$\mathcal{O}(nt)\cdot\lambda$
Shamir [3]	$\mathcal{O}(nt)\cdot\lambda$	$\mathcal{O}(n)\cdot\lambda$	$\mathcal{O}(nt)\cdot\lambda$	$\mathcal{O}(t\log^2 t)\cdot\lambda$
Wang and Desmedt [8]	$\mathcal{O}(n^2)$	$\mathcal{O}(nt)\cdot\lambda$	$\mathcal{O}(nt)\cdot\lambda$	$\mathcal{O}(n)\cdot\lambda^3$

Fig. 3: Comparing bitwise XORs in $(t,n)-$threshold schemes with λ bit secrets.

If pre-computation is impossible and the construction of M in Kurihara et al.'s scheme is ignored, all four schemes have the same distribution complexity.

However, if pre-computation is possible, our scheme has an equivalent complexity to Shamir's scheme and Kurihara et al.'s scheme (again, assuming we ignore the construction of M). Wang and Desmedt's scheme has a heavier complexity and allowing for pre-computation does not decrease the distribution complexity. If the complexity of computing M in Kurihara et al.'s scheme is taken into account, their scheme has by far the heaviest distribution algorithm.

In the best case scenario for our scheme, the recovery complexity is dependent only on t. This is the same as Shamir's scheme but with a lower complexity.

Kurihara et al. discuss how their recovery procedure is dependent on both t and n. This is in contrast to Shamir's scheme, which is dependent only on t. Kurihara et al. say their scheme is more efficient than Shamir's when t is close to n. However, when n is large and t is small, Shamir's is faster. In the worst case scenario for our *Recover* algorithm, the number of bitwise XORs is equivalent to Kurihara et al.'s. Therefore, their argument holds true for our scheme also.

In contrast, Wang and Desmedt's scheme has a complexity dependent only on n, but a factor of λ^3. Being dependent only on n means the scheme will not be able to take advantage of a low threshold value. Also, in many applications, λ may be considerably larger than both n and t, meaning that Wang and Desmedt's scheme may be considerably slower than Kurihara et al.'s scheme and ours.

One other aspect to consider is that of implementation. Our scheme uses a linear $(0, t; n)-$ramp scheme, which is equivalent to a $[n, t, n-t+1]-$MDS code. Rather than using the systematic IDA in our scheme, any MDS code for which there already exists an implementation could be used; as the two are equivalent, the security proof from Section 4 would still hold true. This is also true for Wang and Desmedt's scheme, as their scheme is equivalent to an array code.

6.3 Conclusion

We presented an XOR based $(t, n)-$threshold scheme. We presented a security proof and efficiency analysis and compared our scheme to other schemes with equivalent properties. Our scheme has an equivalent, or more efficient, *Share* complexity to the other schemes considered, especially when pre-computation is possible. The complexity of *Recover* is, in the worst case scenario, equivalent to Kurihara et al.'s scheme, which is faster than Shamir's when t is large. In the best case scenario, our *Recover* complexity is independent of n and faster than all other proposed schemes. Furthermore, our scheme is ideal and, unlike Wang and Desmedt's scheme, requires minimal randomness.

Further work will consider non-ideal $(t, n)-$threshold schemes and whether compromises can be met (with respect to the share size, the number of random bits generated or specifying values for t) that would allow fewer bitwise XORs.

References

1. Chen, L., Camble, P.T., Watkins, M. R., Henry, I.J.: Utilizing error correction (ECC) for secure secret sharing. Hewlett Packard Enterprise Development LP,

World Intellectual Property Organisation Patent Number WO2016048297, https://www.google.com/patents/WO2016048297A1?cl=en (2016)
2. Blakely, G.: Safeguarding cryptographic keys. In: Proc. of the National Computer Conference, vol. 48, pp. 313-317. (1979)
3. Shamir, A.: How to share a secret. In: Communications of the ACM, vol. 22, no. 11, pp. 612-613. ACM (1979)
4. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: A fast (3, n)-threshold secret sharing scheme using exclusive-or operations. In: IEICE transactions on fundamentals of electronics, communications and computer sciences, vol. 91, no. 1, pp. 127-138. IEICE (2008)
5. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: A new (k, n)-threshold secret sharing scheme and its extension. In: International Conference on Information Security, pp. 455-470. Springer (2008)
6. Lv, C., Jia, X., Tian, L., Jing, J., Sun, M.: Efficient ideal threshold secret sharing schemes based on exclusive-or operations. In: 4th International Conference on Network and System Security (NSS), pp. 136-143. IEEE (2010)
7. Lv, C., Jia, X., Lin, J., Jing, J., Tian, L., Sun, M.: Efficient secret sharing schemes. In: FTRA International Conference on Secure and Trust Computing, Data Management, and Application, pp. 114-121. Springer (2011)
8. Wang, Y., Desmedt, Y.: Efficient secret sharing schemes achieving optimal information rate. In: Information Theory Workshop (ITW), pp. 516-520. IEEE (2014)
9. Beimel, A.: Secret-sharing schemes: A survey. In: International Conference on Coding and Cryptology, pp. 11-46. Springer (2011)
10. Jackson, W., Martin, K.M.: A combinatorial interpretation of ramp schemes. In: Australasian Journal of Combinatorics, vol. 14, pp. 52-60. Centre for Combinatorics (1996)
11. MacWilliams, F., Sloane, N.: The theory of error correcting codes. Elsevier (1977)
12. Reed, I., Solomon, G.: Polynomial codes over certain Finite Fields. In: Journal of the society for industrial and applied mathematics, vol. 8, no. 2, pp. 300-304. SIAM (1960)
13. Karnin, E.D., Greene, J.W., Hellman, M.E.: On secret sharing systems. In: Information Theory, IEEE Transactions on, vol. 29, no. 1, pp. 35-41. IEEE (1983)
14. Chen, H., Cramer, R.: Algebraic geometric secret sharing schemes and secure multiparty computations over small fields. In: Annual International Cryptology Conference, pp. 521-536. Springer, Berlin (2006)
15. Krawczyk, H.: Distributed fingerprints and secure information dispersal. In: Proceedings of the twelfth annual ACM symposium on Principles of distributed computing, pp. 207-218. ACM (1993)
16. Rabin, M.: Efficient dispersal of information for security, load balancing, and fault tolerance. In: Journal of the ACM (JACM), vol. 36, no. 2, pp. 335-348. ACM (1989)
17. Resch, J.K., Plank, J.S.: AONT-RS: blending security and performance in dispersed storage systems. In: FAST-2011: 9th USENIX Conference on File and Storage Technologie, pp. 191-202. USENIX Association (2011)
18. Blaum, M., Roth, R.: New array codes for multiple phased burst correction. In: IEEE Transactions on Information Theory, vol. 39, no. 1, pp. 66-77. IEEE (1993)
19. Blundo, C., De Santis, A., Vaccaro, U.: Randomness in distribution protocols. In: Information and Computation, vol. 131, no. 2, pp. 111-139. Elsevier (1996)