A Hierarchical Secret Sharing Scheme Based on Information Dispersal Techniques

Koji Shima and Hiroshi Doi

Institute of Information Security, Japan
dgs164101@iisec.ac.jp, doi@iisec.ac.jp

Abstract. Hierarchical secret sharing schemes are known for how they share a secret among a group of participants partitioned into levels. In this study, we consider using a systematic information dispersal algorithm (IDA). We then apply the general concept of hierarchy to the generator matrix used in a systematic IDA and propose an *ideal* hierarchical secret sharing scheme applicable at any level. For perfect privacy, secret sharing schemes depend on the fact that an adversary can only pool at most $k-1$ shares. However, in our hierarchical scheme, we need to consider an adversary can also pool k or more shares of lower-level participants. Moreover, considering practical use, we present our evaluation of our software implementation.

Keywords: Secret sharing scheme, Hierarchical access structure, IDA, Ideal scheme, Software implementation

1 Introduction

In today's modern information society, there is a strong need to securely store large amounts of secret information and both prevent information theft or leakage and avoid information loss. Secret sharing schemes are known to simultaneously satisfy the need to distribute and manage secret information so as to prevent such information theft and loss. [1] and [2] independently introduced the basic idea of a (k, n) threshold secret sharing scheme almost four decades ago in 1979. In Shamir's (k, n) threshold scheme, n shares are generated from the secret, and each of these shares is distributed to a participant. Next, the secret can be recovered using any subset k of the n shares, but it cannot be recovered with fewer than k shares. Furthermore, every subset comprising less than k participants cannot obtain any information regarding the secret. Therefore, the original secret is secure even if some of the shares are leaked or exposed. Conversely, the secret can be recovered even if a few of the shares are missing.

Several hierarchical secret sharing schemes are known for how they share the given secret among a group of participants who are partitioned into levels. In such schemes, often, a minimal number of higher-level participants are required to recover the secret. For example, opening a bank vault may require, say, three employees, at least one of whom must be a department manager. In this scenario, we have what is called a $(\{1, 3\}, n)$ hierarchical secret sharing scheme. In [3,

4], Tassa introduced polynomial derivatives to generate shares and focused on questions related to Birkhoff interpolation problems.

1.1 Secret Sharing Schemes and Hierarchical Schemes

Kurihara et al. [5–7] proposed $(3, n)$ and (k, n) threshold schemes that use only XOR operations to distribute and recover the secret. In [8], they then presented a faster technique for realizing field operations not over $\mathrm{GF}(q^L)$ but over $\mathrm{GF}(q)$ by using the construction mechanisms of Feng et al. [9] and Blömer et al. [10] for the matrix representation of finite fields. Chen et al. [11] proposed a (k, n) threshold scheme that constructs shares based on a systematic IDA. All above-mentioned schemes are *ideal*.

Next, in [12] and [13], Yamamoto and Blakley et al. each introduced a *ramp* secret sharing scheme, which exhibited a trade-off between security and space efficiency. In [14], Krawczyk proposed a secret sharing scheme called Secret Sharing Made Short, which provides *computational security*, meaning that it encrypts data with a key-based encryption algorithm, then distributes the encrypted data using an IDA and the key via a secret sharing scheme. In [7], Kurihara et al. briefly introduced a *ramp* scheme based on their XOR-based (k, n) threshold scheme. In [15], they then proposed a fast (k, L, n) *ramp* scheme. In [16], Resch et al. proposed a dispersal scheme that provides *computational security*; this scheme enriches Rabin's IDA [17], then combines the All-or-Nothing Transform [18] with the systematic Reed-Solomon code. In [19], Béguin et al. showed how to realize *computational* secret sharing schemes for general access structure. Their approach reduced the problem to an optimization problem.

Tassa [3, 4] proposed a $(\mathbf{k}, n)$ hierarchical secret sharing scheme in which a minimal number of higher-level participants are required for recovering the secret. Tassa's scheme is *ideal*. Tassa used the derivative of a polynomial to achieve hierarchy and recover the secret via Birkhoff interpolation. In [20], Selçuk et al. proposed a function called the truncated version to achieve the described hierarchy. This truncated version truncates the polynomial from to the lowest-order term depending on the level. In [29], Shima et al. proposed a hierarchical secret sharing scheme over finite fields of characteristic two.

In addition, Tassa [3, 4] showed other hierarchical settings studied by other authors. Shamir [2] suggested accomplishing a hierarchical scheme by assigning capable participants a large number of shares. However, when any subset of lower-level participants is sufficiently large, only the lower-level participants are needed to recover the secret. Simmons [21] and Brickell [22] considered other hierarchical settings. However, the necessary number of participants is the highest of the thresholds associated with the various levels. Therefore, their hierarchical settings are unsuitable for the scenario in which a minimal number of higher-level participants must be involved in recovery of the secret.

Tassa then defined a $(\mathbf{k}, n)$ hierarchical secret sharing scheme as follows.

Definition 1 *Let* $\mathbf{k} = \{k_i\}_{i=0}^{m}$, $0 < k_0 < \cdots < k_m$, *and let* $k = k_m$ *be the maximal threshold. A* $(\mathbf{k}, n)$ *hierarchical secret sharing scheme where a minimal*

number of higher-level participants are required for any recovery of the secret is defined as the following access structure Γ:

$$\Gamma = \left\{ \mathcal{V} \subset \mathcal{U} : \left| \mathcal{V} \cap \left(\bigcup_{j=0}^{i} \mathcal{U}_j \right) \right| \geq k_i, \forall i \in \{0, 1, \cdots, m\} \right\}.$$

Here, let $\mathcal{U}$ be a set of n participants and assume that $\mathcal{U}$ is composed of levels, that is, $\mathcal{U} = \bigcup_{i=0}^{m} \mathcal{U}_i$, where $\mathcal{U}_i \bigcap \mathcal{U}_j = \emptyset$ for all $0 \leq i < j \leq m$. The scheme then generates each share of the participants $u \in \mathcal{U}$ to satisfy the access structure.

Given $\mathbf{k} = \{1, 3\}$ as an example, we have a $(\{1, 3\}, n)$ hierarchical scheme that consists of two levels and requires at least one indispensable participant from $\mathcal{U}_0$ and three or more participants from $\mathcal{U}_0 \bigcup \mathcal{U}_1$ to recover the secret.

1.2 Example Scenarios of Hierarchical Schemes

Tassa presented the opening of a bank vault as an example scenario. In this scenario, a fast $(\{1, 3\}, n)$ hierarchical secret sharing scheme is required. Castiglione et al. [23, 24] presented other scenarios; the project manager and team members can access a project workspace according to their levels of authority; nurses may access a subset of patients' clinical data, while a doctor can access all data.

Here, we present a file management system as an example scenario. We store the indispensable participant's share in local storage such as smartphones, and we store the remaining two shares in external storage such as USB mass storage and cloud storage. Only the owner of the smartphone can recover this data by using either of the two external storage devices, and data cannot be recovered using only the two external storage devices. Considering practical use in this scenario, there is a need for a fast $(\{1, 2\}, 3)$ hierarchical secret sharing scheme. In general, a fast $(\{1, k\}, k + 1)$ hierarchical secret sharing scheme would be useful.

1.3 Our Contributions

In this paper, we introduce our hierarchical IDA, which is a hierarchical secret sharing scheme applicable to any level. Our scheme is both *perfect* and *ideal*. We use operations with GF(2^L). For perfect privacy, Chen et al.'s (k, n) threshold scheme [11] depends on the fact that an adversary can only pool at most $k - 1$ shares. However, in our hierarchical scheme, we need to consider an adversary can also pool k or more shares of lower-level participants. Therefore, Chen et al.'s scheme cannot be directly applied to hierarchical schemes, which we present in more detail in Section 4.2. Our overall contributions are summarized as follows:

– We apply hierarchy to the generator matrix used in an IDA and realize a hierarchical secret sharing scheme applicable to any level. We solve the aforementioned issues and provide mathematical proof.

- In a single hierarchy, or a non-hierarchical secret sharing scheme, our scheme is more efficient in implementation than Chen et al.'s scheme [11] because in our scheme, all matrices $\mathbf{G}'$ used by $Recover^{\mathrm{IDA}}$ of the corresponding rows are the same.
- We achieve a $(\{1, k\}, k+1)$ hierarchical scheme using only XOR operations. As a result, we can use simple 64-bit XOR operations instead of $\mathrm{GF}(2^L)$. Then, this scheme is much faster than an approach by Tassa [3, 4].

2 Preliminaries

2.1 Notations and Definitions

We use the following notations and definitions throughout this paper.

- $\oplus$ denotes a bitwise XOR operation.
- $\oplus_{j=a}^{b} c_j$ denotes $c_a \oplus \cdots \oplus c_b$.
- $||$ denotes a concatenation of bit sequences.
- $||_{j=a}^{b} c_j$ denotes $c_a || \cdots || c_b$.
- $H(X)$ denotes the Shannon entropy of a random variable X.
- $\mathbf{v}[j]$ denotes the j-th element in vector $\mathbf{v}$.
- $\mathbf{v}[0][1] \cdots [n-1]$ denotes vector $\mathbf{v}$ with exactly n elements.
- Elements in $\mathrm{GF}(2^L)$ can be identified with polynomials $f_L(X) = \sum_{i=0}^{L-1} f_i X^i$, $f_i \in \mathrm{GF}(2)$. They can also be represented by decimal numbers or hexadecimal numbers of $f_{L-1} \cdots f_1 f_0$ binary. For example, $f_8(X) = X^5 + 1 \in \mathrm{GF}(2)[X]$ can be represented by 33 or 21h of 00100001 binary.

2.2 Perfect and Ideal Secret Sharing Schemes

In this subsection, we refer to Beimel [25] for a *perfect* secret sharing scheme and refer to Blundo et al. [26, 27] and Kurihara et al. [5–7] for an *ideal* secret sharing scheme. Let S be a random variable in a given probability distribution on the secret, S_B be a random variable in a given probability distribution on the shares in each authorized set B, and S_T be a random variable in a given probability distribution on the shares of each unauthorized set T. A *perfect* secret sharing scheme would satisfy the following conditions:

Correctness, Accessibility $H(S|S_B) = 0$.
Perfect privacy, Perfect security $H(S|S_T) = H(S)$.

A secret sharing scheme is called *ideal* if it is *perfect* and the information rate equals one. In other words, if the size of every bit of the shares equals the bit size of the secret, the scheme is *ideal*. As Tassa [4] mentioned in Definition 1.1, we may apply the information rate to a hierarchical secret sharing scheme.

2.3 Systematic IDA

A (k, n) systematic IDA constitutes two more specific algorithms, i.e., $Share^{\mathrm{IDA}}$ and $Recover^{\mathrm{IDA}}$.

$Share^{\mathrm{IDA}}$ takes as input data message D and outputs a codeword to distribute D among n participants. D is parsed into column vector $\mathbf{D}$ that has k elements, with each element in $\mathrm{GF}(2^L)$. Generator matrix or dispersal matrix $\mathbf{G} = [g_{(i,j)}]_{i=1,j=1}^{n\ \ k}$ is a publicly known $n \times k$ matrix with the following conditions:

- The first k rows form the $k \times k$ identity matrix.
- Any subset k of the n rows of $\mathbf{G}$ is linearly independent.

Column vector $\mathbf{C}$ with n elements is then output as codeword $\mathbf{C} = \mathbf{G} \cdot \mathbf{D}$. Since the first k rows of $\mathbf{G}$ form the identity matrix, we obtain

$$\mathbf{C} = \mathbf{G} \cdot \begin{pmatrix} \mathbf{D}[0] \\ \vdots \\ \mathbf{D}[k-1] \end{pmatrix} = \begin{pmatrix} \mathbf{D}[0] \\ \vdots \\ \mathbf{D}[k-1] \\ \mathbf{C}[k] \\ \vdots \\ \mathbf{C}[n-1] \end{pmatrix},$$

where each element $\mathbf{D}[i], \mathbf{C}[i] \in \mathrm{GF}(2^L)$. Then, $\mathbf{G}$ is derived from a Vandermonde matrix using a sequence of elementary matrix transformations. In [28], Plank et al. describe how to prepare an $n \times k$ Vandermonde matrix in which $g_{(i,j)} = (i-1)^{j-1}$ and we turn the first k rows into the identity matrix using a sequence of elementary matrix transformations. This satisfies the conditions of $\mathbf{G}$ since elementary matrix operations do not change the rank of a matrix. Furthermore, a square Vandermonde matrix with $g_{(i,j)} = x^{j-1}$ is invertible if all x are distinct.

$Recover^{\mathrm{IDA}}$ takes as input the remaining k elements $\mathbf{C}'$ for codeword $\mathbf{C}$ and outputs data message D. Here, $\mathbf{C}'$ is a column vector that has k elements. Through this process, new $k \times k$ matrix $\mathbf{G}'$ is formed from $\mathbf{G}$ and corresponds to the remaining k elements. After $\mathbf{G}'$ is inverted, we obtain D via $\mathbf{D} = (\mathbf{G}')^{-1} \cdot \mathbf{C}'$.

Employing a (k, n) systematic IDA instead of a (k, n) non-systematic IDA improves performance because it does not need to encode the first k codeword elements and similarly does not need to decode codeword elements that are equal to message data elements.

3 Related Work

From [11], Chen et al. presented a scheme that constructs an *ideal* threshold scheme with a systematic IDA. Since an IDA is essentially a *ramp* scheme, their scheme generates dummy keys at random, passing both these dummy keys and the secret values XORed with these dummy keys to the systematic IDA. Their scheme then applies some cyclic shifts to each of the outputs to generate shares.

Let P_x for $x = 0, \cdots, n-1$ be n participants for the (k, n) threshold scheme over $F = \mathrm{GF}(2^L)$. Then, generator matrix $\mathbf{G}$ is publicly known and has elements in $\mathrm{GF}(2^L)$, as remarked in Section 3.3. Furthermore, let the secret be given by $s \in \{0,1\}^\lambda, \lambda = L \cdot k$. Secret s is parsed into $\mathbf{s} \in F^k$ with k elements of length L bits. Here, s must be padded to λ bits if s is less than λ bits.

3.1 Chen et al.'s Distribution Algorithm

Table 1 shows their distribution algorithm. Step 1 generates random values called dummy keys $r_1, \cdots, r_{k-1} \in \{0,1\}^\lambda$. These values are parsed into $\mathbf{r}_1, \cdots, \mathbf{r}_{k-1} \in F^k$. In Step 2, s and the dummy keys are XORed to produce $s' \in \{0,1\}^\lambda$, then s' is parsed into $\mathbf{r}_0 \in F^k$. In Step 3, each $\mathbf{r}_i$ is passed into $Share^{\mathrm{IDA}}$. As a result, we obtain each column vector $\mathbf{R}_0, \cdots \mathbf{R}_{k-1} \in F^n$, each of which has n elements. In Steps 4 and 5, each participant P_x securely receives share $w_x \in \{0,1\}^\lambda$. To shed further light on this algorithm, we detail Steps 3 and 4. In Step 3, we illustrate $k \times n$ matrix

$$\mathbf{M} = \begin{pmatrix} \mathbf{R}_0^{\mathrm{T}} \\ \mathbf{R}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{R}_{k-1}^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} \mathbf{R}_0[0][1] \cdots [n-1] \\ \mathbf{R}_1[0][1] \cdots [n-1] \\ \vdots \\ \mathbf{R}_{k-1}[0][1] \cdots [n-1] \end{pmatrix}.$$

Next, in Step 4, we illustrate matrix

$$\mathbf{M}' = \begin{pmatrix} \mathbf{R}_0[0]\ [1]\ \cdots\ [n-2]\ [n-1] \\ \mathbf{R}_1[1]\ [2]\ \cdots\ [n-1]\ \ [0] \\ \vdots\ \ \vdots\ \ \ \ \ \vdots\ \ \ \ \vdots \\ \mathbf{R}_{k-1}[k-1]\ [k]\ \cdots\ [k-3]\ [k-2] \end{pmatrix},$$

applying $j-1$ left cyclic shifts to elements of the j-th row of $\mathbf{M}$ for $j = 1, \cdots, k$. Further, w_x concatenates the elements in the $x+1$-th column of $\mathbf{M}'$.

3.2 Chen et al.'s Recovery Algorithm

Table 2 shows the corresponding recovery algorithm. The algorithm takes as input shares of participants P_i for $i = t_0, \cdots, t_{k-1}$ that cooperate to recover the secret. Step 1 parses each participant's share w_i into its k elements. More specifically, k of the n elements for each row are given in the distribution algorithm. Since elements in each row are cyclically shifted when the shares are generated, indexes of the k elements are different in each row, implying that all matrices $\mathbf{G}'_i$ passed into $Recover^{\mathrm{IDA}}$ for the corresponding row differ. In Step 2, each column vector $\mathbf{R}'_i$ for $i = 0, \cdots, k-1$ has the k elements of the $i+1$-th row in Step 1, and these column vectors are passed into $Recover^{\mathrm{IDA}}$. In Steps 3 and 4, s' and $r_1, \cdots, r_{k-1}$ are then recovered from the available shares. Finally, in Steps 5 and 6, these recovered values are XORed to retrieve secret s.

Table 1. Distribution algorithm

Input: $s \in \{0,1\}^{\lambda}$
Output: $(w_0, \cdots w_{n-1})$
1: for $i \leftarrow 1$ to $k-1$:
$\quad \mathbf{r}_i \leftarrow r_i \xleftarrow{\$} \{0,1\}^{\lambda}$
2: $\mathbf{r}_0 \leftarrow s' \leftarrow s \oplus \{\oplus_{j=1}^{k-1} r_j\}$
3: for $i \leftarrow 0$ to $k-1$:
$\quad \mathbf{R}_i \leftarrow Share^{\mathrm{IDA}}(\mathbf{r}_i, \mathbf{G})$
4: for $i \leftarrow 0$ to $n-1$:
$\quad w_i \leftarrow \|_{j=0}^{k-1} \mathbf{R}_j[i+j \pmod n]$
5: return $(w_0, \cdots w_{n-1})$

Table 2. Recovery algorithm

Input: $(w_{t_0}, \cdots w_{t_{k-1}})$
Output: s
1: for $i \leftarrow 0$ to $k-1$:
$\quad \|_{j=0}^{k-1} \mathbf{R}_j[t_i + j \pmod n] \leftarrow w_{t_i}$
2: for $i \leftarrow 0$ to $k-1$:
$\quad \mathbf{r}_i \leftarrow Recover^{\mathrm{IDA}}(\mathbf{R}'_i, \mathbf{G}'_i)$
3: for $i \leftarrow 0$ to $k-1$:
$\quad r_i \leftarrow \mathbf{r}_i$
4: $s' \leftarrow r_0$
5: $s \leftarrow s' \oplus \{\oplus_{j=1}^{k-1} r_j\}$
6: return s

3.3 Remark

Chen et al. showed generator matrix $\mathbf{G}$ as an $n \times k$ binary matrix. Any subset k of the n rows of $\mathbf{G}$ should be linearly independent, but not all of the parameters with k and n can satisfy the condition. Given $k = 3$ and $n = 5$ as an example, we cannot find any combinations of $g_0, g_1, g_2 \in \mathrm{GF}(2)$ in

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ g_0 & g_1 & g_2 \end{pmatrix}.$$

In general, $\mathbf{G}$ has elements in $\mathrm{GF}(2^L)$. With $n = k, k+1$, $\mathbf{G}$ has elements in $\mathrm{GF}(2)$.

4 Our Proposed Scheme

In this section, we describe our proposed $(\mathbf{k}, n)$ hierarchical secret sharing scheme that satisfies Definition 1. We use operations with $F = \mathrm{GF}(2^L)$. Let the secret be given by $s \in \{0,1\}^{\lambda}, \lambda = L \cdot k$. Secret s is parsed into $\mathbf{s} \in F^k$ with k elements of length L bits. Note that s must be padded to λ bits if s is less than λ bits. We use $n \times k$ generator matrix $\mathbf{G}$ with the following properties:

- $\mathbf{G}$ has a defined hierarchy such that only an authorized subset can recover the secret.
- $\mathbf{G}$ does not have any row of $(y \ \cdots \ y)$, where $y \in F$.

$\mathbf{G}$ is publicly known. Since it is a uniquely determined table in a fixed system, we are able to recover the secret only from shares. We may describe $\mathbf{G}$ before using elementary matrix transformations required for the systematic IDA. This $\mathbf{G}$ is hereinafter referred to as a hierarchical generator matrix, and the IDA using this $\mathbf{G}$ is also hereinafter referred to as a hierarchical IDA.

4.1 Participant Identities and Hierarchical Generator Matrix

Let $P_x \in \mathcal{U}$ for $x = 0, \cdots, n-1$ define n participants and let $0 \leq l_0 \leq \cdots \leq l_m = n$. Each participant P_x has identity $x \in F$. Without loss of generality, we may assume that each P_x belongs to the following levels:

$$P_0, \cdots, P_{l_0-1} \in \mathcal{U}_0, \quad P_{l_0}, \cdots, P_{l_1-1} \in \mathcal{U}_1, \quad \cdots, \quad P_{l_{m-1}}, \cdots, P_{l_m-1} \in \mathcal{U}_m.$$

For example, $l_0 = 0$ means no participants belong to $\mathcal{U}_0$. Furthermore, let $0 \in \mathcal{U}_0$ be the phantom participant and let u_x (described later) always be assigned to zero.

P_x corresponds to the $x+1$-th row of $\mathbf{G}$. In other words, we can view the $n \times k$ matrix as $\mathbf{G} = [g_{(x,j)}]_{x=0,\ j=1}^{n-1\ \ k}$. Then, we introduce a hierarchy to $\mathbf{G}$, constructing $\mathbf{G}$ such that $k \times k$ matrix $\mathbf{G}'$ satisfies $\det(\mathbf{G}') = 0$ for any unauthorized k participants, where $\mathbf{G}'$ is generated from the rows of $\mathbf{G}$ corresponding to the k participants. Here, $P_x \in \mathcal{U}_i$ for $i = 0, \cdots, m$ generates $\mathbf{G}$ with

$$g_{(x,j)} = \begin{cases} u_x^{j-1-k_{i-1}} & (j > k_{i-1}) \\ 0 & (j \leq k_{i-1}) \end{cases},$$

where $g_{(x,j)} \in F$ and $k_{-1} = 0$. Given a $(\{2,3,5\}, n)$ hierarchical secret sharing scheme as an example, we obtain

$$\mathbf{G} = \begin{pmatrix} 1 & u_0 & u_0^2 & u_0^3 & u_0^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & u_{l_0-1} & u_{l_0-1}^2 & u_{l_0-1}^3 & u_{l_0-1}^4 \\ 0 & 0 & 1 & u_{l_0} & u_{l_0}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & u_{l_1-1} & u_{l_1-1}^2 \\ 0 & 0 & 0 & 1 & u_{l_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & u_{n-1} \end{pmatrix}.$$

Intuitively, shares of lower-level participants are not generated from the secret and some random values. Here, u_x for $\mathbf{G}$ can be assigned from $2^L - 1$ values except zero over F. However, depending on the assignment of u_x, there is a specific case in which the secret cannot be recovered in spite of the presence of an authorized subset. Given $\mathbf{G}$ for a $(\{1,3\}, 5)$ hierarchical scheme over $\mathrm{GF}(2^8)$ as an example to understand the meaning of $\det(\mathbf{G}') = 0$, one of the matrices $\mathbf{G}_1^{'}$ derives $\det(\mathbf{G}_1^{'}) = 0$ when $\mathbf{G}_1$ is given by $u_x = 1, 2, 3, 4, 5$. $\mathbf{G}_2$ given by $u_x = 1, 2, 4, 5, 6$ is required. Note that elements are represented by decimal numbers following Section 2. It is sufficient to find one $\mathbf{G}$, such as $\mathbf{G}_2$ for this scheme, i.e.,

$$\mathbf{G}_1 = \begin{pmatrix} 1\ 1\ 1 \\ 1\ 2\ 4 \\ 0\ 1\ 3 \\ 0\ 1\ 4 \\ 0\ 1\ 5 \end{pmatrix}, \mathbf{G}_1^{'} = \begin{pmatrix} 1\ 1\ 1 \\ 1\ 2\ 4 \\ 0\ 1\ 3 \end{pmatrix}, \mathbf{G}_2 = \begin{pmatrix} 1\ 1\ 1 \\ 1\ 2\ 4 \\ 0\ 1\ 4 \\ 0\ 1\ 5 \\ 0\ 1\ 6 \end{pmatrix}.$$

Therefore, there is a case in which $\det(\mathbf{G}') = 0$, where the corresponding $\mathbf{G}'$ to the authorized subset. As Tassa described in Section 3.2 of [4] regarding probability, we can make the same argument for this issue. In other words, Tassa stated that the p used to allocate participant identities over GF(p) should be usually very large. Similarly, the L used to allocate u_x of $\mathbf{G}$ over GF(2^L) can be large in our scheme. To keep the paper compact, we do not discuss it further.

4.2 An Issue with Applying Hierarchy to IDA

A (k, n) IDA recovers not only data message D, but also all n elements if any k elements are given because the remaining $n - k$ elements can be calculated from publicly known generator matrix $\mathbf{G}$. A $(\mathbf{k}, n)$ hierarchical IDA also has a similar property. We consider a $(\{1, 3\}, 8)$ hierarchical IDA as an example. Let d_1, d_2, d_3 be passed into the systematic hierarchical IDA and let the eight codeword elements be $d_1, d_2, d_3, c_1, \cdots, c_5$. Furthermore, let d_1, d_2 be used for $\mathcal{U}_0$ and let $d_3, c_1, \cdots, c_5$ be used for $\mathcal{U}_1$. If three elements in $\mathcal{U}_1$ are given, such as c_1, c_2, c_3, we need to consider all elements not only in $\mathcal{U}_1$ but also in $\mathcal{U}_0$ can be calculated. As we described in Section 1.3, secret sharing schemes depend on the fact that an adversary can only pool at most $k - 1$ shares. In Chen et al.'s scheme, the cyclic shifts after $Share^{\mathrm{IDA}}$ work well to satisfy perfect privacy. However, in our hierarchical scheme, the cyclic shifts after $Share^{\mathrm{IDA}}$ have no effect to satisfy perfect privacy because we need to consider no elements in $\mathcal{U}_0$ can be calculated when three elements in $\mathcal{U}_1$ are given in the example.

4.3 Distribution Algorithm

The dealer securely distributes each share $w_x \in \{0, 1\}^\lambda$ to participant P_x. Table 3 shows this specific algorithm. The underlined portions show differences between the algorithm of Table 1 and our algorithm. More specifically, with each column vector $\mathbf{r}_0, \cdots, \mathbf{r}_{k-1}$ transposed, Step 3 constructs matrix

$$\mathbf{M}_r = \begin{pmatrix} \mathbf{r}_0[0] & \cdots & \mathbf{r}_0[k-1] \\ \vdots & \ddots & \vdots \\ \mathbf{r}_{k-1}[0] & \cdots & \mathbf{r}_{k-1}[k-1] \end{pmatrix} = \begin{pmatrix} \mathbf{r}'_0[0] & \cdots & \mathbf{r}'_{k-1}[0] \\ \vdots & \ddots & \vdots \\ \mathbf{r}'_0[k-1] & \cdots & \mathbf{r}'_{k-1}[k-1] \end{pmatrix}$$

and defines $\mathbf{r}'_0, \cdots, \mathbf{r}'_{k-1}$ as column vectors. In Step 5, no cyclic shifts are used after $Share^{\mathrm{IDA}}$. Through Steps 1 through 5 of our algorithm, we can illustrate $k \times n$ matrix

$$\mathbf{M} = \begin{pmatrix} \mathbf{R}_0^{\mathrm{T}} \\ \vdots \\ \mathbf{R}_{k-1}^{\mathrm{T}} \end{pmatrix} = \left(\begin{array}{ccc|c|ccc} \overbrace{\mathbf{R}_0[0] & \cdots & [l_0-1]}^{\mathcal{U}_0} & \cdots & \overbrace{[l_{m-1}] & \cdots & [n-1]}^{\mathcal{U}_m} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \mathbf{R}_{k-1}[0] & \cdots & [l_0-1] & \cdots & [l_{m-1}] & \cdots & [n-1] \end{array}\right).$$

Each participant P_x securely receives share w_x concatenating the elements in the $x + 1$-th column of $\mathbf{M}$.

Table 3. Our distribution algorithm

Input: $s \in \{0,1\}^\lambda$
Output: $(w_0, \cdots w_{n-1})$
1: for $i \leftarrow 1$ to $k-1$:
$\quad \mathbf{r}_i \leftarrow r_i \xleftarrow{\$} \{0,1\}^\lambda$
2: $\mathbf{r}_0 \leftarrow s' \leftarrow s \oplus \{\oplus_{j=1}^{k-1} r_j\}$
3: $\mathbf{M}_r = (\mathbf{r}'_0 \ \cdots \ \mathbf{r}'_{k-1}) \leftarrow (\mathbf{r}_0 \ \cdots \ \mathbf{r}_{k-1})^{\mathrm{T}}$
4: for $i \leftarrow 0$ to $k-1$:
$\quad \mathbf{R}_i \leftarrow \mathit{Share}^{\mathrm{IDA}}(\underline{\mathbf{r}'_i, \mathbf{G}})$
5: for $i \leftarrow 0$ to $n-1$:
$\quad w_i \leftarrow \|_{j=0}^{k-1} \mathbf{R}_j \underline{[i]}$
6: return $(w_0, \cdots w_{n-1})$

Table 4. Our recovery algorithm

Input: $(w_{t_0}, \cdots w_{t_{k-1}})$
Output: s
1: for $i \leftarrow 0$ to $k-1$:
$\quad \|_{j=0}^{k-1} \mathbf{R}_j \underline{[t_i]} \leftarrow w_{t_i}$
2: for $i \leftarrow 0$ to $k-1$:
$\quad \underline{\mathbf{r}'_i} \leftarrow \mathit{Recover}^{\mathrm{IDA}}(\mathbf{R}'_i, \underline{\mathbf{G}'})$
3: $\underline{(\mathbf{r}_0 \ \cdots \ \mathbf{r}_{k-1})^{\mathrm{T}} \leftarrow \mathbf{M}_r = (\mathbf{r}'_0 \ \cdots \ \mathbf{r}'_{k-1})}$
4: for $i \leftarrow 0$ to $k-1$:
$\quad r_i \leftarrow \mathbf{r}_i$
5: $s' \leftarrow r_0$
6: $s \leftarrow s' \oplus \{\oplus_{j=1}^{k-1} r_j\}$
7: return s

4.4 Recovery Algorithm

Table 4 shows our recovery algorithm. This algorithm takes as input shares of participants P_i for $i = t_0, \cdots, t_{k-1}$ that cooperate to recover the secret. The underlined portions show differences between the algorithm of Table 2 and our algorithm. Since there are no cyclic shifts after $\mathit{Share}^{\mathrm{IDA}}$, all matrices $\mathbf{G}'$ used by $\mathit{Recover}^{\mathrm{IDA}}$ of the corresponding rows are the same. Step 1 parses each participant's share w_i into its k elements. In Step 2, each column vector $\mathbf{R}'_i$ for $i = 0, \cdots, k-1$ has the k elements of the $i+1$-th row from Step 1, and these column vectors are passed into $\mathit{Recover}^{\mathrm{IDA}}$. In Steps 3 through 5, s' and $r_1, \cdots, r_{k-1}$ are recovered from the available shares. Finally, in Steps 6 and 7, these recovered values are XORed to retrieve secret s.

4.5 Security Analysis

Theorem 1 proves the correctness and perfect privacy of our proposed scheme. We obtain secret s if all elements of $\mathbf{M}_r$ are recovered with $\mathbf{G}'$. Without loss of generality, we may focus on k elements of each j-th row of $\mathbf{M}_r^{\mathrm{T}}$, recovered from the j-th row of $\mathbf{M}$ with $\mathit{Recover}^{\mathrm{IDA}}$, because each j-th row can be processed independently from the others. We then apply this argument to every other row.

Next, the j-th row of $\mathbf{M}_r^{\mathrm{T}}$ has $k-1$ random values $\mathbf{r}'_{j-1}[1], \cdots, \mathbf{r}'_{j-1}[k-1] \in F$ and the value $\mathbf{r}'_{j-1}[0]$ XORed with those random values and secret $\mathbf{s}[j-1] \in F$. Therefore, any $k-1$ of the k values cannot reveal anything about the secret. We then pass the k values into $\mathit{Share}^{\mathrm{IDA}}$ and obtain $\mathbf{R}_{j-1}$. Lemma 1 proves that $\mathbf{R}_{j-1}$ cannot reveal anything about the secret.

Lemma 1. *Assume that any set of L' participants $T = \{P_{t_0}, \cdots, P_{t_{L'-1}}\} \notin \Gamma$ agrees to recover the secret. Let $y \in F \backslash \{0\}$. Then, if $n \times k$ generator matrix $\mathbf{G}$ whose j-th row is $(y \ \cdots \ y)$ is not used, we receive no information regarding the secret. In information theoretic terms, $H(S|S_T) = H(S)$, shown in Section 2.2.*

Proof. Suppose that s and $r_1, \cdots, r_{k-1}$ are mutually independent and that $r_1, \cdots, r_{k-1}$ are selected from the finite set $\{0,1\}^\lambda$ with uniform probability $1/2^\lambda$. We define $k \times k$ matrices $\mathbf{X}$, $\mathbf{A}$, and $\mathbf{A}'$ as

$$\mathbf{X} = \begin{pmatrix} 1\ 1 \cdots 1 \\ 0\ 1 \cdots 0 \\ \vdots\ \vdots\ \ddots\ \vdots \\ 0\ 0 \cdots 1 \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} \mathbf{s}^{\mathrm{T}} \\ \mathbf{r}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{r}_{k-1}^{\mathrm{T}} \end{pmatrix}, \quad \mathbf{A}' = \begin{pmatrix} 0 \ \cdots \ 0 \\ \mathbf{r}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{r}_{k-1}^{\mathrm{T}} \end{pmatrix},$$

respectively. Here, $\mathbf{X}^{-1} = \mathbf{X}$. Steps 1 through 3 of Table 3 can be represented as $\mathbf{M}_r = \mathbf{X} \cdot \mathbf{A}$. With $Share^{\mathrm{IDA}}$, we obtain $n \times k$ matrix $\mathbf{M}^{\mathrm{T}} = \mathbf{G} \cdot \mathbf{M}_r$. We then define $n \times k$ matrix $\mathbf{Y} = \mathbf{G} \cdot \mathbf{X}$, we obtain

$$\mathbf{M}^{\mathrm{T}} = \mathbf{G} \cdot \mathbf{M}_r = \mathbf{G} \cdot \mathbf{X} \cdot \mathbf{A} = \mathbf{Y} \cdot \mathbf{A}.$$

Here, let $\mathbf{G}_{(i)}$ be a matrix constructed by any subset i of the n rows of $\mathbf{G}$ for $i = 1, \cdots, L'$ and let $\mathbf{Y}_{(i)} = \mathbf{G}_{(i)} \cdot \mathbf{X}$. Furthermore, let $\mathbf{M}_{(i)}^{\mathrm{T}} = \mathbf{Y}_{(i)} \cdot \mathbf{A}$. Without loss of generality, we may assume that $\mathrm{rank}(\mathbf{G}_{(i)}) = i$ because we can consider $\mathbf{G}_{(\alpha)}$ such that $\mathrm{rank}(\mathbf{G}_{(i)}) = \alpha < i$. Then, $\mathrm{rank}(\mathbf{Y}_{(i)}) = \mathrm{rank}(\mathbf{G}_{(i)})$ because $\mathbf{X}$ is regular. Consider $i = 1$. It is apparent that the corresponding participant receives share $w \in \{0,1\}^\lambda$ regarding secret s if $\mathbf{Y}_{(1)} = (y\ 0\ \cdots\ 0)$, i.e.,

$$w \leftarrow \mathbf{M}_{(1)}^{\mathrm{T}} = \mathbf{Y}_{(1)} \cdot \mathbf{A} = \left(y \cdot \mathbf{s}^{\mathrm{T}}\right), \quad \mathbf{G}_{(1)} = \mathbf{Y}_{(1)} \cdot \mathbf{X}^{-1} = \left(y \cdots y\right).$$

If $\mathbf{Y}_{(1)} \neq (y\ 0\ \cdots\ 0)$, the vector $\mathbf{Y}_{(1)} \cdot \mathbf{A}'$ is uniformly distributed over $\{0,1\}^\lambda$ because all elements of the vector are L-bit random numbers that are mutually independent and distributed uniformly over $\{0,1\}^L$. Then, we suppose w' denotes a fixed value of w. $w = w'$ can be obtained with uniform probability $1/2^\lambda$ from any selected s. Because s is independent of w, we have $H(S|S_T) = H(S)$.

Next, we prove that vector $(y, 0, \cdots, 0)$ is not expressed by linear combination of the row vectors of $\mathbf{Y}_{(i)}$ for $i \geq 2$. Here, we may assume that none of the rows of $\mathbf{G}_{(i)}$ are equal to $(y\ \cdots\ y)$ because we already prove $\mathbf{G}_{(1)} = (y\ \cdots\ y)$. Matrix

$$\mathbf{Y}_{(i)} = \begin{pmatrix} y_1\ a_{(1,1)} \cdots a_{(1,k-1)} \\ \vdots\ \ \vdots\ \ \ddots\ \ \vdots \\ y_i\ a_{(i,1)} \cdots a_{(i,k-1)} \end{pmatrix} = \mathbf{G}_{(i)} \cdot \mathbf{X}$$

can be represented. If $y_j = 0$ for $j = 1, \cdots, i$, we receive no information regarding the secret because the j-th row of $\mathbf{Y}_{(i)}$ is formed by $(0\ *)$. Consider $y_j \neq 0$ for all $j = 1, \cdots, i$. There exist matrices $\mathbf{A}_{(i)}$ and $\mathbf{B}_{(i)}$ such that

$$\mathbf{G}_{(i)} = \mathbf{Y}_{(i)} \cdot \mathbf{X}^{-1} = \begin{pmatrix} y_1 \cdots y_1 \\ \vdots\ \ddots\ \vdots \\ y_i \cdots y_i \end{pmatrix} + \begin{pmatrix} 0\ a_{(1,1)} \cdots a_{(1,k-1)} \\ \vdots\ \ \vdots\ \ \ddots\ \ \vdots \\ 0\ a_{(i,1)} \cdots a_{(i,k-1)} \end{pmatrix} = \mathbf{A}_{(i)} + \mathbf{B}_{(i)}.$$

Since rank is subadditive and $\mathrm{rank}(\mathbf{A}_{(i)}) = 1$,

$$\mathrm{rank}(\mathbf{A}_{(i)} + \mathbf{B}_{(i)}) \leq \mathrm{rank}(\mathbf{A}_{(i)}) + \mathrm{rank}(\mathbf{B}_{(i)}) = 1 + \mathrm{rank}(\mathbf{B}_{(i)})$$

and the rank of $\mathbf{B}_{(i)}$ is either $i-1$ or i. If $\mathrm{rank}(\mathbf{B}_{(i)}) = i$, $\sum_{j=1}^{i} a_j \cdot a_{(j,t)} \neq 0$ for at least one of $t = 1, \cdots k-1$, where $a_1, \cdots a_i$ are scalars. Therefore, the vector expressed by linear combination of the row vectors of $\mathbf{Y}_{(i)}$ is not $(y, 0, \cdots, 0)$. Next, if $\mathrm{rank}(\mathbf{B}_{(i)}) = i-1$, there exists matrix $\mathbf{G}_{(i)}$ of rank $i-1$. Furthermore, the remaining row of $\mathbf{G}_{(i)}$ should be assigned for the highest level $\mathcal{U}_0$ because $y_j \neq 0$. However, in our scheme, rows of $\mathbf{G}$ in the highest level are derived from a Vandermonde matrix. In such a condition, $\mathrm{rank}(\mathbf{G}_{(i)})$ should be i, i.e., $\mathrm{rank}(\mathbf{B}_{(i)}) \neq i-1$ by proof by contradiction. The proof is thus complete. □

Theorem 1. *Assume that corresponding square matrix $\mathbf{M}_{\mathcal{V}}$, or namely, the $\mathbf{G}'$ required to recover the secret, is regular for any minimal authorized subset $\mathcal{V} \in \Gamma$, i.e., $|\mathcal{V}| = k$. Then both correctness and perfect privacy hold.*

Proof. Theorem 1 is based on an approach by Tassa [3, 4]. We first consider correctness. Each participant P_x receives a part of the share $\sigma(x)_j$ generated by elements of the j-th row of $\mathbf{M}_r^{\mathrm{T}}$ that are passed into the IDA. Here, let $\mathbf{G}(x)$ denote the $x+1$-th row of $\mathbf{G}$. We then obtain $\sigma(x)_j = \mathbf{G}(x) \cdot \mathbf{r}'_{j-1}$. When all participants $\mathcal{V} = \{P_{t_0}, \cdots, P_{t_{k-1}}\}$ pool their shares $\boldsymbol{\sigma}_j$ together, they need to solve $\boldsymbol{\sigma}_j = \mathbf{M}_{\mathcal{V}} \cdot \mathbf{r}'_{j-1}$, i.e.,

$$\boldsymbol{\sigma}_j = \begin{pmatrix} \sigma(t_0)_j \\ \vdots \\ \sigma(t_{k-1})_j \end{pmatrix} = \begin{pmatrix} \mathbf{G}(t_0) \\ \vdots \\ \mathbf{G}(t_{k-1}) \end{pmatrix} \begin{pmatrix} \mathbf{r}'_{j-1}[0] \\ \vdots \\ \mathbf{r}'_{j-1}[k-1] \end{pmatrix} = \mathbf{G}' \cdot \mathbf{r}'_{j-1}$$

in unknown vector $\mathbf{r}'_{j-1}$. Since $\mathbf{M}_{\mathcal{V}}$ is regular by the assumption noted above, we are able to uniquely solve unknown vector $\mathbf{r}'_{j-1}$ for every j-th row of $\mathbf{M}_r^{\mathrm{T}}$.

Next, we consider perfect privacy. For the j-th row of $\mathbf{M}_r^{\mathrm{T}}$, we may view $\mathbf{r}'_{j-1}[0]$ in the unknown vector $\mathbf{r}'_{j-1}$ as a secret. Furthermore, a square matrix is regular if and only if its determinant is nonzero. Equivalently, the rows of $\mathbf{M}_{\mathcal{V}}$ are linearly independent. Let $\mathcal{V}_u \notin \Gamma$ be an unauthorized subset and $\mathbf{M}_{\mathcal{V}_u}$ be the corresponding matrix. We aim to show that even if all participants in $\mathcal{V}_u$ pool their shares together, they cannot reveal anything regarding the secret. This also implies that any value of the secret is accepted from their shares. The proof here is that the secret is not included in the row space of $\mathbf{M}_{\mathcal{V}_u}$, in the set of all possible linear combinations of the rows of $\mathbf{M}_{\mathcal{V}_u}$.

Without loss of generality, we may assume that $\mathcal{V}_u$ is missing only one participant to become authorized, and we may simplify the process by adding to $\mathcal{V}_u$ phantom participant $0 \in \mathcal{U}_0$ such that we can obtain an authorized subset. Then the square matrix corresponding to the authorized subset is regular by the assumption, and the rows of the square matrix are linearly independent. As a result, the share of participant $0 \in \mathcal{U}_0$ cannot be generated from $\mathcal{V}_u$, and the share is equivalent to $\mathbf{r}'_{j-1}[0]$. In addition, even when $\mathcal{V}_u$ is missing only one participant at the j-th level, the access structure holds by adding one higher-level participant, i.e., the highest-level participant $0 \in \mathcal{U}_0$.

The proof is thus complete and we conclude that $\det(\mathbf{M}_{\mathcal{V}}) \neq 0$ is required for both correctness and perfect privacy. □

5 Software Implementation

We evaluated our scheme using one general purpose machine, as described in Table 5. We then used a file size of 888,710 bytes as an example. For operations

Table 5. Test environment

CPU / RAM	Intel ® Celeron ® Processor G1820
	2.70 GHz × 2, 2 MB cache / 3.6 GB
OS	CentOS 7 Linux 3.10.0-229.20.1.el7.x86_64
Programing language / Compiler	C / gcc 4.8.3 (-O3 -flto -DNDEBUG)

with GF(2^L), the additive operation is replaced by the XOR operation, the multiplication operation uses the Russian peasant multiplication method, the division operation uses $x^{-1} = x^{2^L-2}$, and the shift operation uses only the shift operation by one bit. In our experiments, we only used GF(2^8) and a lookup table that was precomputed for the multiplication and division operations over GF(2^8). More specifically, all results of the multiplication operations were prestored in an array of 2^{16} bytes, while those for the division operations were stored in another array of 2^{16} bytes. When each of the multiplication and division operations actually took place, the operation consisted of a lookup in each array. No cryptographic libraries were used. Then, the primitive polynomial used for GF(2^8) is $x^8 + x^4 + x^3 + x^2 + 1$. Table 6 shows our experimental results.

Table 6. Results of our experiments for recovery

Level **k**	Number of participants ($\|\mathcal{U}_0\|, \cdots, \|\mathcal{U}_m\|$)	Throughput (Mbps)
{1,3}	$(2, 3)$	857
{2,4}	$(3, 4)$	562
{2,3,5}	$(3, 3, 3)$	373
{2,4,6,10}	$(2, 3, 6, 4)$	108
{3,7,11,14,17}	$(3, 4, 5, 4, 4)$	34.9

In terms of optimization, we can construct a scheme using only XOR operations with $(k + 1) \times k$ binary generator matrix

$$\mathbf{G} = \begin{pmatrix} 1\,0 \cdots 0 \\ 0\,1 \cdots 0 \\ \vdots\ \vdots\ \ddots\ \vdots \\ 0\,0 \cdots 1 \\ 0\,1 \cdots 1 \end{pmatrix}.$$

This matrix requires no multiplication operations. We then used simple 64-bit XOR operations. With $k = 3$, we achieved approximately 6.3 Gbps for recovery.

5.1 Computational Costs

Tassa's Approach Table 7 shows the computational costs of the recovery algorithm. In general, the size of the secret, for example, a file size of 1MB, exceeded L bits. In our analysis, we refer to such an initial computation processed once for that recovery as a precomputation. Here, a $t \times t$ determinant required $\mathcal{O}(t^3)$ when we used LU decomposition. Converting a matrix to a triangular matrix required some division operations. Furthermore, a $t \times t$ matrix inverse also required $\mathcal{O}(t^3)$ and some division operations when we used Gaussian elimination. Finally, $t \times t$ matrix multiplication required $\mathcal{O}(t^3)$ if carried out naively. Note that Tassa's scheme can be applied only to GF(p), where p is a prime number. Arithmetic operations, using multiple-precision arithmetic, required higher computational costs than operations over GF(2^L). Therefore, our scheme was much faster than Tassa's approach.

Table 7. Computational costs for recovery

Tassa's approach

Precomputation	One $k \times k$ determinant
Recovery per time	One $k \times k$ determinant, one division operation

Our scheme

Precomputation	One $k \times k$ matrix inverse
Recovery per time	One $k \times k$ matrix product, $k-1$ XOR operations

Blömer et al.'s Technique We consider applying Blömer et al.'s technique to our scheme because Kurihara et al. [8] reported that any secret sharing scheme over GF(2^L) could be converted to a scheme over GF(2). As Blömer et al. [10] mentioned in Section 4, we stored all coefficient vectors of field elements in a table and used table look-ups to compute τ. This operation required $\mathcal{O}(1)$. Furthermore, matrix-vector multiplication required at most L XOR operations. In our implementation using a lookup table for the multiplication operation over GF(2^8), a multiplication operation only required $\mathcal{O}(1)$.

6 Conclusions

In this paper, we focused on a fast $(\mathbf{k}, n)$ hierarchical secret sharing scheme applicable to any level. To achieve this, we applied a hierarchy to the generator matrix used in an IDA. Our scheme is both *perfect* and *ideal*. Given the implementation used in our experiments applicable to any level, we found our implementation on a general purpose PC was able to recover the given secret with $\mathbf{k} = \{1, 3\}$ at a processing speed of approximately 850 Mbps. With our $(\{1, 3\}, 4)$ optimized scheme, we achieved approximately 6.3 Gbps for recovery.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments. This work was supported by JSPS KAKENHI Grant Number JP18K11306.

References

1. Blakley, G.R.: Safeguarding cryptographic keys. AFIPS, Vol.48, 313–317 (1979)
2. Shamir, A.: How to share a secret. Commun. ACM, Vol.22, No.11, 612–613 (1979)
3. Tassa, T.: Hierarchical Threshold Secret Sharing. TCC 2004, LNCS 2951, 473–490 (2004)
4. Tassa, T.: Hierarchical Threshold Secret Sharing. Journal of Cryptology, Vol.20, No.2, 237–264 (2007)
5. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: A Fast $(3, n)$-Threshold Secret Sharing Scheme Using Exclusive-OR Operations. IEICE Trans. Fundamentals, Vol.E91-A, No.1, 127–138 (2008)
6. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: On a Fast (k, n)-Threshold Secret Sharing Scheme. IEICE Trans. Fundamentals, Vol.E91-A, No.9, 2365–2378 (2008)
7. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: A New (k, n)-Threshold Secret Sharing Scheme and Its Extension. ISC 2008, LNCS 5222, 455–470 (2008)
8. Kurihara, J., Uyematsu, T.: A Novel Realization of Threshold Schemes over Binary Field Extensions. IEICE Trans. Fundamentals, Vol.E94-A, No.6, 1375–1380 (2011)
9. Feng, G.-L., Deng, R.-H., Bao, F.: Packet-loss resilient coding scheme with only XOR operations. IEE Proc. Communications, Vol.151, No.4, 322–328 (2004)
10. Blömer, J., Kalfane, M., Karp, R., Karpinski, M., Luby, M., Zuckerman, D.: An XOR-Based Erasure-Resilient Coding Scheme. ICSI Technical Report TR-95-048 (1995)
11. Chen, L., Laing, T.M., Martin K.M.: Efficient, XOR-Based, Ideal (t, n) threshold Schemes. CANS 2016, LNCS 10052, 467–483 (2016)
12. Yamamoto, H.: On Secret Sharing System Using (k, L, n) Threshold Scheme. IEICE Trans. Fundamentals (Japanese Edition), Vol.J68-A, No.9, 945–952 (1985); Secret sharing system using (k, L, n) threshold scheme. Electronics and Communications in Japan (English Edition), Part I, Vol.69, No.9, 46–54 (1986)
13. Blakley, G.R., Meadows C.: Security of Ramp Schemes. Advances in Cryptology - CRYPTO '84, LNCS 196, 242–268 (1985)
14. Krawczyk, H.: Secret Sharing Made Short. Advances in Cryptology - CRYPTO '93, LNCS 773, 136–146 (1993)
15. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: A Fast (k, L, n)-Threshold Ramp Secret Sharing Scheme. IEICE Trans. Fundamentals, Vol.E92-A, No.8, 1808–1821 (2009)
16. Resch, J.K., Plank, J.S.: AONT-RS: blending security and performance in dispersed storage systems. In 9th USENIX Conference on File and Storage Technologie, 191–202 (2011)
17. Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. Journal of the ACM, Vol.36, No.2, 335–348 (1989)
18. Rivest, R.L.: All-or-nothing encryption and the package transform. FSE 1997, LNCS 1267, 210–218 (1997)

19. Béguin P., Cresti A.: General Short Computational Secret Sharing Schemes. Advances in Cryptology - EUROCRYPT '95, LNCS 921, 194–208 (1995)
20. Selçuk, A.A., Kaşkaloğlu, K., Özbudak, F.: On Hierarchical Threshold Secret Sharing. IACR Cryptology ePrint Archive 2009, 450 (2009)
21. Simmons, G.J.: How to (really) share a secret. Advances in Cryptology - CRYPTO '88, LNCS 403, 390–448 (1990)
22. Brickell, E.F.: Some ideal secret sharing schemes. Advances in Cryptology - EUROCRYPT '89, LNCS 434, 468–475 (1990)
23. Castiglione, A., De Santis, A., Masucci, B.: Hierarchical and shared key assignment. NBiS-2014, pp.263–270 (2014)
24. Castiglione, A., De Santis, A., Masucci, B., Palmieri, F., Castiglione, A., Li, J., Huang, X.: Hierarchical and Shared Access Control. IEEE Trans. Information Forensics and Security, Vol.11, No.4, pp.850–865 (2016)
25. Beimel, A.: Secret-Sharing Schemes, A Survey. IWCC 2011, LNCS 6639, 11–46 (2011)
26. Blundo, C., De Santis, A., Gargano, L., Vaccaro, U.: On the information rate of secret sharing schemes. TCS, Vol.154, 283–306 (1996)
27. Blundo, C., De Santis, A., Gargano, L., Vaccaro, U.: On the Information Rate of Secret Sharing Schemes. Advances in Cryptology - CRYPTO '92, LNCS 740, 149–169 (1993)
28. Plank, J.S., Ding, Y.: Note: Correction to the 1997 tutorial on Reed-Solomon coding. Software - Practice & Experience, Vol.35, No.2, 189–194 (2005)
29. Shima, K., Doi, H.: A Hierarchical Secret Sharing Scheme over Finite Fields of Characteristic 2. Journal of Information Processing, Vol.25, pp.875–883 (2017)