博士論文

Research on Fast Hierarchical Secret Sharing Schemes

Koji SHIMA
島　幸司

情報セキュリティ大学院大学
情報セキュリティ研究科
情報セキュリティ専攻

2019年3月

Research on Fast Hierarchical Secret Sharing Schemes

Koji Shima

A dissertation
Submitted to the Department of Information Security
of the Graduate School of Information Security
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in Informatics

INSTITUTE of INFORMATION SECURITY, JAPAN
2019

Abstract

Hierarchical secret sharing schemes (HSSSs) are known for how they share a secret among a group of participants partitioned into levels. In this research, we aim to propose fast HSSSs in which a minimal number of higher-level participants are required for recovery of the secret.

In the modern information society, there is a strong need to securely store large amounts of secret information to prevent information theft or leakage and avoid information loss. Secret sharing schemes (SSSs) are known to simultaneously satisfy the need to distribute and manage secret information to prevent such information theft and loss. Blakley and Shamir independently introduced the basic idea of a (k, n) threshold SSS 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.

Shamir suggested accomplishing a hierarchical scheme by assigning capable participants a large number of shares at the same time. In Shamir's approach, when any subset of lower-level participants is sufficiently large, only the lower-level participants are needed to recover the secret. However, in HSSSs, 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 $(\{1, 3\}, n)$ HSSS. Tassa introduced polynomial derivatives to generate shares and focused on the question related to Birkhoff interpolation problems.

Given that the hierarchical scheme requires indispensable participants to recover a

given secret, from another perspective, this scheme can be used to delete the secret. In (k, n) threshold schemes, the reliability of data deletion depends on the deletion of more than $n - k$ shares. However, while using such a hierarchical scheme, the reliability of data deletion depends on the deletion of the specific shares held by the indispensable participants. Considering practical and strategic use, for example, in cases where data deletion must be done with urgency, deletion of the secret is guaranteed by the deletion of the shares possessed by the indispensable participants. In other words, this scheme satisfies the need to prevent both information theft and information loss. Moreover, it offers the advantage of facilitating quick and straightforward deletion of the secret after it has been distributed.

In this research, we propose four HSSSs. First, we present a HSSS over finite fields of characteristic two through Tassa's HSSS of using derivatives and Birkhoff interpolation. When we differentiate a polynomial $p(x)$ over finite fields of characteristic two, every term that is an even degree will disappear since differentiating x^i such that i is an even number results in zero. Given $p(x) = a_2x^2 + a_1x + a_0 \in \mathrm{GF}(2^L)[x]$, where $a_2, a_1, a_0 \in \mathrm{GF}(2^L)$, as an example, we obtain $p'(x) = a_1$ and $p''(x) = 0$. For that reason, Tassa's scheme does not work as expected over finite fields of characteristic two. Our scheme provides the missing piece for the case of characteristic two, in which meaningful shares cannot be generated as long as we apply Tassa's scheme as is to the finite fields of characteristic two because the k-th derivative of polynomial $p(x)$, where $k \geq 2$, always results in $p^{(k)}(x) = 0$. We introduce a new technique called the n-th order reduction of $p(x)$. As a result, the Birkhoff interpolation works with modification where the derivative of a polynomial is replaced with our function. Furthermore, since Tassa's scheme can be applied only to $\mathrm{GF}(p)$, where p is a prime number, arithmetic operations, using multiple-precision arithmetic, requires higher computational costs than operations over $\mathrm{GF}(2^L)$. Therefore, our scheme is much faster than Tassa's approach.

Next, we apply the general concept of hierarchy to the generator matrix used in a systematic information dispersal algorithm (IDA) and propose a HSSS that can be applied to any level. Chen et al. proposed a (k, n) threshold SSS that constructs shares based on a systematic IDA. An SSS depends on the fact that an adversary can only pool at most $k - 1$ shares. Their SSS also depends on this fact. However, in our hierarchical scheme, we need to consider an adversary can also pool k or more shares of lower-level participants. Therefore, their scheme cannot be directly applied to hierarchical schemes. In our HSSS, we solve this issue and provide mathematical proof. In a non-hierarchical

SSS, our scheme is more efficient in implementation than their SSS. Furthermore, 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 GF(2^L). Then, this scheme is much faster than an approach by Tassa.

Finally, we present XOR-based HSSSs. More specifically, we propose a $(\{1, 3\}, n)$ XOR-based HSSS for a small number of indispensable participants specially made for three participants including one or two indispensable participants to recover the secret through Fujii et al.'s SSS. In our HSSS, we pass a random number into Fujii et al.'s $(2, n)$ threshold SSS to generate intermediate shares and mix the secret only for each of the intermediate shares of the indispensable participants. Next, we consider applying the general concept of hierarchy to Kurihara et al.'s XOR-based SSS and propose a HSSS that can be applied to any level. When k participants at the highest level cooperate to recover the secret, the proposed scheme can yield the same result as Kurihara et al.'s (k, n) threshold SSS because our scheme requires a minimal number of higher-level participants to recover the secret.

In such hierarchical schemes, including Tassa's scheme, depending on the allocation of participant identities, there is a specific case in which the secret cannot be recovered in spite of the presence of an authorized subset. We consider this allocation issue. Furthermore, considering practical use, we present each evaluation of our software implementation and discuss the findings. Given that the implementation used in our experiments can be applied to any level, we found that our implementation was able to recover a given secret with $(\{1, 3\}, n)$ at a processing speed of approximately 97 Mbps with any n for the first approach, 850 Mbps with any n for the second approach, and 8.0 Gbps, taken as the average from additional experimental results, with $n = 5$ for the fourth approach. Then, under the implementation optimized for a $(\{1, 3\}, n)$ HSSS for recovery, we achieved approximately 970 Mbps with any n for the first approach, 6.3 Gbps with $n = 4$ for the second approach, and 8.3 Gbps with $n = 5$ for the third approach. We then conclude that we can use our schemes depending on the situation.

Acknowledgments

It has been a great honor and pleasure to study under the mentorship of Professor Hiroshi Doi for these five years, including two-year master's course. Prof. Doi gave me this great opportunity when he agreed to supervise me three years ago. As an adviser, he patiently and skillfully guided me, always recommending what direction I should proceed in. Moreover, he has shared with me his large experience and deep perspective of the field. I appreciate all that he has done for me.

Besides my advisor, it must have been somewhat difficult to review this dissertation. I offer my sincere thanks to Professor Seiko Arita, Professor Takao Okubo, and Professor Akira Otsuka for their support and invaluable advice. I would also like to thank Professor Naoshi Sato, who guided me during my training.

I would like to thank Doi laboratory mates for their fruitful discussions. I would also like to thank other laboratory mates, especially Arita laboratory mates for making my graduate school experience exciting and fun. Furthermore, I would like to thank those who have offered their help and provided their comments at ISS square, symposiums (CSS 2015, CSS 2016, CSS 2017, and SCIS 2018), AsiaJCIS 2016, IWSEC 2018, ICISC 2018, IPSJ awards ceremony in 2018, and study groups (CSEC 72 and ISEC on March 2018). Then, I would also like to thank anonymous reviewers for their helpful comments.

I would like to thank my coworkers and managers. It would have been impossible for me to balance work and study without their understanding and cooperation. In particular, I would like to express my gratitude to Kenichi Imai and Yoshikazu Onuki. It would have been impossible to start my study without their understanding of entering Institute of Information Security. I would also like to express my gratitude to Taichi Okabayashi. My study would never have reached completion without their understanding.

Last but not least, I would like to express my deepest gratitude to my family. This dissertation would not have been possible without their continued support and encouragement.

Contents

CHAPTER 1

Introduction

In the modern information society, there is a strong need to securely store large amounts of secret information to prevent information theft or leakage and avoid information loss. Secret sharing schemes (SSSs) are known to simultaneously satisfy the need to distribute and manage secret information to prevent such information theft and loss. Blakley [1] and Shamir [2] independently introduced the basic idea of a (k, n) threshold SSS 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 (HSSSs) 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 $(\{1, 3\}, n)$ HSSS. In [8, 9], Tassa introduced polynomial derivatives to generate shares and focused on the question related to Birkhoff interpolation problems.

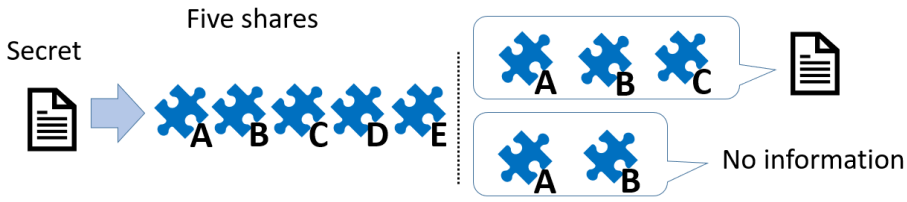
Given that the hierarchical scheme described above requires indispensable participants to recover a given secret, from another perspective, this scheme can be used to

delete the secret. In (k, n) threshold schemes, the reliability of data deletion depends on the deletion of more than $n-k$ shares. However, while using such a hierarchical scheme, the reliability of data deletion depends on the deletion of the specific shares held by the indispensable participants. Considering practical and strategic use, for example, in cases where data deletion must be done with urgency, deletion of the secret is guaranteed by the deletion of the shares possessed by the indispensable participants. In other words, this scheme satisfies the need to prevent both information theft and information loss. Moreover, it offers the advantage of facilitating quick and straightforward deletion of the secret after it has been distributed.

1.1 Secret Sharing Schemes

Shamir's (k, n) threshold SSS [2] is both *perfect* and *ideal*, and it can be implemented using arbitrary values of k and n, with $k \leq n$. Figure 1.1 shows a $(3, 5)$ threshold SSS as an example. However, the scheme requires extensive calculations for generating

Figure 1.1: $(3, 5)$ threshold SSS



the n shares and recovering the secret from k shares because in doing so, a polynomial of degree $k-1$ must be processed. Fujii et al. [11] proposed a fast $(2, n)$ threshold scheme that uses only XOR operations to distribute and recover the secret. Kurihara et al. [12, 13, 14] proposed a $(3, n)$ threshold scheme that uses only XOR operations and a (k, n) threshold scheme that uses only XOR operations. Following up on their own work and the fact that the computational cost of both multiplication and division over GF(2^L) is high compared to that of the additive approach, in [15], they presented a faster technique for realizing field operations not over GF(q^L) but over GF(q) by using the construction mechanisms of Feng et al. [16] and Blömer et al. [17] for the matrix representation of finite fields. Chen et al. [18] proposed a fast (k, n) threshold scheme that constructs shares based on a systematic information dispersal algorithm (IDA). All

above-mentioned schemes are *ideal*.

Next, in [40] and [41], Yamamoto and Blakley et al. each introduced a *ramp* SSS, which exhibited a trade-off between security and space efficiency. In [42], Krawczyk proposed an SSS called Secret Sharing Made Short (SSMS), 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 an SSS. In [14], Kurihara et al. briefly introduced a *ramp* scheme based on their XOR-based (k, n) threshold scheme. In [27], they then proposed a fast (k, L, n) *ramp* scheme. In [43], Resch et al. proposed a dispersal scheme that provides *computational security*; this scheme enriches Rabin's IDA, then combines the All-or-Nothing Transform (AONT) [44] with the systematic Reed-Solomon (RS) code; therefore, this scheme is called AONT-RS. In [45], Béguin et al. showed how to realize *computational* secret sharing schemes for general access structure. Their approach reduced the problem to an optimization problem. Finally, in [46], Ikarashi et al. proposed a fast SSS based on SSMS with a fast multiplication technique over GF(2^{64}) in terms of studies and practical use of erasure codes.

Matsuo et al. [29] presented a technique using XOR-operations. Suga [30, 31, 32], Ke et al. [33], and Ozaki et al. [34] presented their studies and proposals, respectively. Cianciullo et al. [35] proposed schemes related to cheating detection capability. Secret sharing with cheating detection capability considers the scenario in which cheaters modify their shares to trick other honest participants into reconstructing a false secret.

1.2 Hierarchical Secret Sharing Schemes

Shamir [2] suggested accomplishing a hierarchical scheme by assigning capable participants a large number of shares. Here, levels are represented as subsets of participants, but the necessary number of participants for recovery is determined based on a weighted average of the thresholds associated with each of these levels. In other words, when any subset of lower-level participants is sufficiently large, only the lower-level participants are needed to recover the secret. Kothari [3] considered hierarchical schemes in which a simple (k_i, n_i) threshold scheme is associated with the i-th level of a multilevel group. As Ghodosi et al. mentioned in [7], this solution does not provide concurrency among different levels of hierarchical groups.

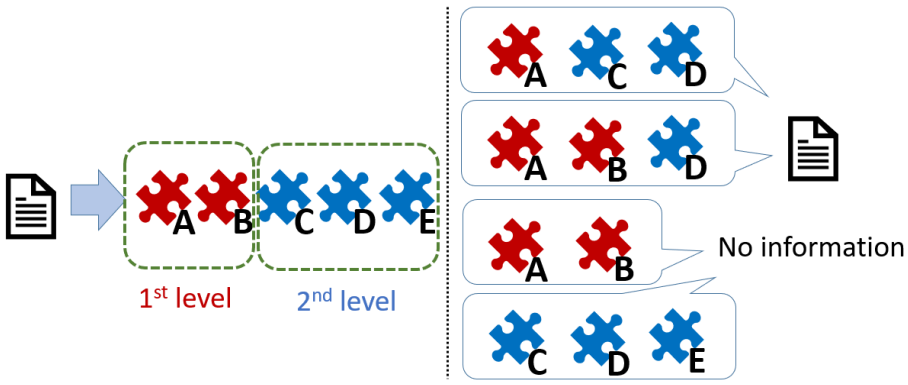
Ito et al. [4, 5, 6] first studied secret sharing for general access structures, called multiple assignment scheme. They proved that every access structure can be realized

by a *perfect* scheme. As Ghodosi et al. [7] mentioned, their scheme may assign the same share to many participants. In other words, more privileged participants are given longer shares. As Tassa [8, 9] also mentioned, the resulting schemes for threshold access structures are far from being *ideal*.

Simmons [20] studied the problem with a *disjunction* of the threshold conditions. As Tassa [8, 9] mentioned, his solution is based on a geometric construction presented by Blakley [1]. Then, the scheme is not *ideal*. Brickell [21] offered two *ideal* disjunctive HSSSs. As Ghodosi et al. [7] mentioned, however, constructing efficient solution to these schemes was left as an open problem. Later, they [7] presented efficient solutions. In Simmons [20] and Brickell [21], 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. Furthermore, in Ghodosi et al. [7], as they mentioned in their bank example, there is a case in which only the lower-level participants can recover the secret.

Tassa [8, 9] proposed a $(\mathbf{k}, n)$ *ideal* HSSS in which a minimal number of higher-level participants are required for recovering the secret. We have what is called a conjunctive HSSS. Figure 1.2 shows a $(\{1, 3\}, 5)$ conjunctive HSSS as an example. Furthermore,

Figure 1.2: $(\{1, 3\}, 5)$ HSSS



Tassa focused on questions related to Birkhoff interpolation problems. He used the derivative of a polynomial to achieve hierarchy and recover the secret via Birkhoff interpolation. Kurihara et al. [10] presented constructions for XOR-based conjunctive HSSS and disjunctive HSSS. Their scheme achieved hierarchy by generating shares of

lower-level participants without the secret and some random values and cannot yield the same result as Kurihara et al.'s (k, n) threshold SSS [13] when k participants at the highest level cooperate to recover the secret. Their scheme then has no mathematical proof for perfect privacy. Farràs et al. [28] solved what hierarchical access structures admit an *ideal* secret sharing scheme. Selçuk et al. [19] proposed a function called the truncated version to achieve a conjunctive HSSS. This truncated version truncates the polynomial from to the lowest-order term depending on the level.

Käsper [38] investigated the *multiplicativity* of hierarchical schemes. Multiplicativity allows participants, holding shares of two secrets s_0 and s_1, to privately compute shares of the product $s_0 s_1$ without revealing the original secrets. Roy et al. [39] presented definitions and constructions of both cheater identifiable and robust HSSSs.

1.3 HSSS Example Scenarios

Given that HSSSs require indispensable participants to recover a given secret, from another perspective, HSSSs can be used to delete the secret. In (k, n) threshold SSSs, the reliability of data deletion depends on the deletion of more than $n - k$ shares. However, while using such a hierarchical scheme, the reliability of data deletion depends on the deletion of the specific shares held by the indispensable participants. Considering practical and strategic use, for example, in cases where data deletion must be done with urgency, deletion of the secret is guaranteed by the deletion of the shares possessed by the indispensable participants. In other words, this scheme satisfies the need to prevent both information theft and information loss. Moreover, it offers the advantage of facilitating quick and straightforward deletion of the secret after it has been distributed.

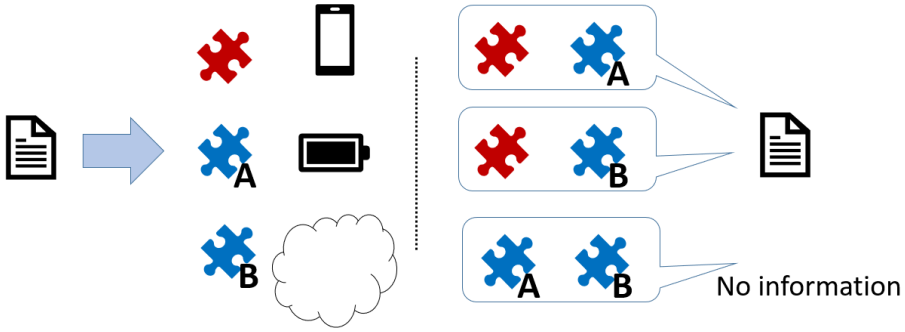
Considering a HSSS over $\mathrm{GF}(2^L)$ and a HSSS using only XOR operations, we can achieve a smaller calculation load and can realize fast HSSSs. Therefore, such HSSSs, which can be applied to any level, are required.

Tassa [8, 9] presented the opening of a bank vault as an example scenario. In this scenario, a fast $(\{1, 3\}, n)$ HSSS is required in terms of optimization. Castiglione et al. [22, 23] 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 consider other two scenarios. One is a file management system. 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 $(\{1,2\},3)$ HSSS. Figure 1.3 shows the example. The other is a secure cloud system.

Figure 1.3: $(\{1,2\},3)$ HSSS file management system



Matsumoto et al. [47] considered the need to reduce server costs in the practical implementations of their SSS; accordingly, a smaller number of servers would be required. Therefore, there is a need for a fast scheme such as a $(\{1,3\},4)$ HSSS. In the two aforementioned scenarios, a fast $(\{1,k\},k+1)$ HSSS would be useful as well in terms of optimization.

1.4 Fast Schemes

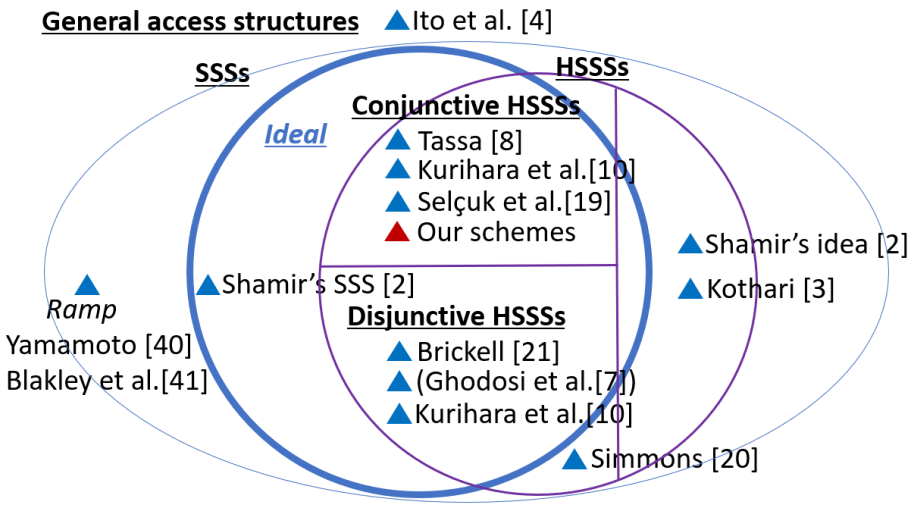
Operations over GF(2^L) yield fast schemes, but one multiplication operation requires a few XOR operations, as in the case of Kurihara et al.'s analysis [15]. This computational cost can be one operation if we use a lookup table that has been precomputed for the multiplication operation over GF(2^L); otherwise, there remains little choice but to practically choose $L = 8$ in terms of the amount of available memory. To extend the size of L, Ikarashi et al.'s technique [46] is suited for fast multiplication operations over GF(2^{64}). Because their fast technique requires an extended CPU instruction set, it cannot be applied to all hardware, such as embedded devices, which are used widely in the Internet of Things. Schemes constructed using simple XOR operations can use the maximum bit length of XOR, such as 64 bits. In other words, only simple XOR

operations yield faster schemes.

1.5 Relations of SSSs

Figure 1.4 gives relations of SSSs. In secret sharing for general access structures, not all

Figure 1.4: Overall picture and our contributions



of the schemes are *ideal* and provide concurrency among different levels of hierarchical groups. HSSSs provide concurrency among different levels, such as a scheme in which a minimal number of higher-level participants are required for recovering the secret, but not all HSSSs are also *ideal*. SSSs have only one level, but some SSSs can be specially made by a HSSS with one level.

1.6 Our Overall Contributions

Considering the HSSS example scenarios, shown in Section 1.3, and the advantage of facilitating quick and straightforward deletion of the secret after it has been distributed, we aim to propose fast HSSSs.

In Chapter 3, we present a HSSS that can be applied to any level over finite fields of characteristic two through Tassa's HSSS of using derivatives and Birkhoff interpolation. Our scheme provides the missing piece for the case of characteristic two which Tassa's

scheme [8, 9] lacks. Our scheme then has a firm mathematical basis. More specifically, the Birkhoff interpolation works with modification where the derivative of a polynomial is replaced with our function. In Chapter 4, we introduce our hierarchical IDA, which is a HSSS that can be applied to any level. We solve several issues and provide mathematical proof. In a single hierarchy, or a non-hierarchical SSS, our scheme is more efficient in implementation than Chen et al.'s scheme [18] because in our scheme, all matrices $\mathbf{G}'$ used by $Recover^{\mathrm{IDA}}$ of the corresponding rows are the same. We then 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 GF(2^L). This scheme is much faster than an approach by Tassa [8, 9]. In Chapter 5, we propose a $(\{1, 3\}, n)$ XOR-based HSSS for a small number of indispensable participants through Fujii et al.'s threshold SSS. We pass a random number into Fujii et al.'s $(2, n)$ threshold SSS to generate intermediate shares and mix the secret only for each of the intermediate shares of the indispensable participants. In Chapter 6, we apply the general concept of hierarchy to Kurihara et al.'s XOR-based SSS and realize a simple XOR-based HSSS, which can be applied to any level. When k participants at the highest level cooperate to recover the secret, the proposed scheme can yield the same result as Kurihara et al.'s (k, n) threshold SSS because our scheme requires a minimal number of higher-level participants to recover the secret. All above-mentioned schemes are both *perfect* and *ideal*. They then are conjunctive HSSSs, in which a minimal number of higher-level participants are required for recovering the secret.

In such hierarchical schemes, depending on the allocation of participant identities, there is a specific case in which the secret cannot be recovered in spite of the presence of an authorized subset. We need to further discuss this, but achieving hierarchy with the proposed XOR-based scheme in Chapter 6 could eliminate the traditional allocation issue.

In Chapter 7, considering practical use, we present each evaluation of our software implementation and discuss the findings. We then conclude that we can use our schemes depending on the situation.

CHAPTER 2

Preliminaries

In this chapter, we prepare for definitions required in this dissertation.

2.1 Notations and Definitions

Throughout this dissertation, we use the following notations and definitions.

- $n \in \mathbb{N}$ denotes number of participants.
- $\oplus$ 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.
- $\xleftarrow{\$} \mathcal{X}$ denotes a function to generate an $|\mathcal{X}|$-bit random number from a finite set $\mathcal{X}$.
- 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.

Chapter 4 uses the following notations and definitions.

- $\mathbf{v}[j]$ denotes the j-th element in vector $\mathbf{v}$.
- $\mathbf{v}[0][1]\cdots[n-1]$ denotes vector $\mathbf{v}$ with exactly n elements.

Chapters 5 and 6 use the following notations and definitions.

- n_p is a prime number such that $n_p \geq n$.
- The index values of (1) random numbers, (2) divided pieces of the secret and the shares, (3) XOR-ed terms of (1) and (2), (4) participants, and (5) matrices are elements of GF(n_p), that is, $X_{c(a\pm b)}$ denotes $X_{c(a\pm b) \bmod n_p}$.

2.2 Definitions and Properties of Matrix

These are used only in Chapter 6.

- $\mathbf{I}_n$ denotes the $n \times n$ identity matrix.
- $\boldsymbol{O}$ denotes a zero matrix.
- $\mathbf{1}$ denotes a matrix in which all elements are ones.
- Elementary row operations are as follows: (1) Each element in a row is multiplied by a non-zero constant, (2) A row is replaced by the sum of that row and a multiple of another row, (3) A row within the matrix is switched with another row. Note that these are informal explanations of row operations.
- The leading coefficient of a row is the first nonzero element in that row.
- A matrix in row echelon form is one in which the leading coefficient of a nonzero row is strictly to the right of the leading coefficient of the row above, and all nonzero rows are above any rows of all zeros.
- Any matrix can be transformed to the row echelon form by using elementary row operations. The echelon form is not unique, but all row echelon forms have the same number of zero rows.
- The rank of matrix $\mathbf{A}$, $rank(\mathbf{A})$, equals the number of nonzero rows of a matrix transformed to the row echelon form.

2.3 Basic Theorem of Matrix

The linear system $\mathbf{Ax} = \mathbf{b}$, with $m \times n$ coefficient matrix $\mathbf{A}$ and $m \times (n+1)$ augmented matrix $[\mathbf{A}\ \mathbf{b}]$, admits solutions if and only if

$$rank(\mathbf{A}) = rank([\mathbf{A}\ \mathbf{b}]).$$

If $rank(\mathbf{A}) = n$, the solution is unique. If $rank(\mathbf{A}) < n$, the system has an infinite number of solutions and has $n - rank(\mathbf{A})$ degrees of freedom. Moreover, there exists a unique solution when we select particular values for $n - rank(\mathbf{A})$ free variables, which correspond to the columns of $\mathbf{A}$ without leading coefficients.

2.4 Perfect SSS

In [24], Beimel showed in Definitions 2 and 3 that a *perfect* SSS requires the following conditions:

Correctness, Accessibility Every authorized set B in an access structure receives the secret information.

Perfect privacy, Perfect security Every unauthorized set T outside of any access structure receives no secret information.

In other words, 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* SSS would satisfy the following conditions:

Correctness, Accessibility $H(S|S_B) = 0$.

Perfect privacy, Perfect security $H(S|S_T) = H(S)$.

2.5 Ideal SSS

In this section, we refer to Blundo et al. [25, 26] and Kurihara et al. [12, 13, 14]. Let $\mathcal{P} = \{P_1, \cdots, P_n\}$ be a set of n participants. The dealer selects secret $s \in \mathcal{S}$ and distributes each share $w_i \in \mathcal{W}_i$ to participant $P_i \in \mathcal{P}$, where $\mathcal{S}$ denotes the set of secrets

and $\mathcal{W}_i$ denotes the set of possible shares that P_i might receive. The information rate is then defined as $\rho = \frac{H(S)}{\max\limits_{P_i \in \mathcal{P}} H(W_i)}$, where S and W_i denote the random variables induced by $s \in \mathcal{S}$ and $w_i \in \mathcal{W}_i$, respectively. When the probability distributions over $\mathcal{S}$ and shares $\mathcal{W}_i$ are uniform, the information rate is

$$\rho = \frac{\log_2 |\mathcal{S}|}{\max\limits_{P_i \in \mathcal{P}} \log_2 |\mathcal{W}_i|}.$$

An SSS is called *ideal* if it is *perfect* and $\rho = 1$. 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 [9] mentioned in Definition 1.1, we may apply the information rate to a HSSS.

2.6 Conjunctive HSSS

Tassa [8, 9] defined a $(\mathbf{k}, n)$ HSSS 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 $(\mathbf{k}, n)$ HSSS 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.

We require in Definition 1 a *conjunction* of threshold conditions. Given $\mathbf{k} = \{1, 3\}$ as an example, we have a $(\{1, 3\}, n)$ HSSS that consists of two levels in the hierarchy 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.

2.7 Disjunctive HSSS

Simmons [20] and Brickell [21] studied the problem with a *disjunction* of the threshold conditions. The access structure is defined by the replacement of the universal quantifier

$\forall$ in Definition 1 with the existential quantifier $\exists$, i.e.,

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

2.8 Ramp SSS

Below, we use the same definition as [18] for the (k, L, n) *ramp* scheme [40]. We also refer to Jackson et al. [48].

Definition 2. *A $(t_0, t_1; n)$ ramp SSS is a method of distributing a secret such that any set of at least t_1 participants can pool their shares to uniquely recover the secret. A set of t_0 or fewer participants reveals no information about the secret.*

This definition reveals that a $(k-1, k; n)$ *ramp* scheme is a (k, n) threshold scheme. It also shows that a $(0, k; n)$ *ramp* scheme has no perfect privacy and has only correctness for any k participants. Furthermore, if a $(t_0, t_1; n)$ *ramp* scheme is *linear*, every further share reveals $\frac{1}{t_1 - t_0}$ bits of information about the secret after t_0 shares are pooled.

2.9 Information Dispersal Algorithm

From [49], Rabin's (k, n) IDA breaks file F of length $|F|$ into n pieces of length $|F|/k$ and reconstructs F with any k pieces. The IDA is space-efficient since n/k can be chosen to be close to one. In other words, the total size of n pieces is $n|F|/k$. Furthermore, since an IDA has no requirements for perfect privacy and has only correctness [18, 42], a (k, n) IDA is equivalent to a $(0, k; n)$ *ramp* scheme [18].

2.10 Error-Correcting Code and Erasure Code

An error-correcting code is a method of encoding a message into a codeword with some redundant or parity data so as to ensure that the message can be recovered even if errors to the capability of the code occur in either data transmission or storage. The code is *systematic* if every codeword can be explicitly broken down into a message data

part and a redundant data part. Since IDAs can be implemented in a variety of ways, all corresponding to the notion of erasure codes in error-correcting code theory [42], an erasure code is equivalent to a (k, n) IDA [18].

2.11 Test Environment

We evaluated our schemes and other related schemes using one general purpose machine, as described in Table 2.1. Table 2.1 also shows the GCC options related to performance. We then used a file size of 888,710 bytes as an example and provided some parameters of $\mathbf{k}$ and n.

Table 2.1: Test environment

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

For operations with GF(2^L) in Chapters 3 and 4, 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 pre-stored 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. Then, no cryptographic libraries were used. Finally, the primitive polynomial used for GF(2^8) is $x^8 + x^4 + x^3 + x^2 + 1$.

CHAPTER 3

HSSS over Finite Fields of Characteristic Two

In this chapter, we present a HSSS that can be applied to any level over finite fields of characteristic two, published at [55] [1] , through Tassa's idea of using derivatives and Birkhoff interpolation. Our contribution can be summarized as follows:

- Our scheme provides the missing piece for the case of characteristic two which Tassa's scheme [8, 9] lacks. As we will see in more detail in Section 3.3, meaningful shares cannot be generated as long as we apply Tassa's scheme as is to the finite fields of characteristic two because the k-th derivative of polynomial $p(x)$ always results in $p^{(k)}(x) = 0$ where $k \geq 2$. We introduce a new technique.

- Our scheme has a firm mathematical basis. More specifically, the Birkhoff interpolation works with modification where the derivative of a polynomial is replaced with our function, shown in Section 3.4.1. Note that our function is different from the truncated version, developed by Selçuk et al. [19].

[1] [53] is the preliminary version of [55]. [53] was published at Computer Security Symposium 2016 (CSS 2016) and was recommended to be submitted to Journal of Information Processing (JIP) by the program chair of the symposium. [55] was selected as Specially Selected Paper by the editorial committee of the JIP at the IPSJ, Information Processing Society of Japan (⟨http://www.ipsj.or.jp/english/organization/aboutipsj/award/ssp_award.html⟩). We also received JIP Outstanding Paper Award (⟨http://www.ipsj.or.jp/award/ronbun-index.html⟩). [61] is the report published in reports of the 2017 IPSJ Best Paper Award.

- Our scheme achieves a high throughput due to the binary operations in characteristic two. Taking practical and strategic use into consideration, a $(\{1, k\}, n)$ HSSS will be useful as well, especially with $k = 3$. Our scheme also achieves the same effect of [52].

3.1 Tassa's HSSS

Tassa's scheme is both *perfect* and *ideal*. The scheme does not allow the lower-level participants alone to recover the secret. Furthermore, the secret is the free coefficient or constant term of some polynomial $p(x)$ of degree $k - 1$ over a large finite field in the same manner as in Shamir's (k, n) threshold scheme. Each participant $u \in \mathcal{U}$ is given an identity in the field, also denoted by u, and a share that equals $p^{(j)}(u)$ for some derivative order j that depends on the position of u in the hierarchy. The more important participants belong to levels with a lower index and will therefore receive shares with lower derivative orders. We can construct the access structure by selecting appropriate derivative orders.

3.2 Polynomial Interpolation

When we use a polynomial interpolation to recover the secret instead of solving the simultaneous equations, we contribute to a smaller calculation load. However, when a derivative value is included as a share, the secret cannot be recovered with either Lagrange interpolation or Newton's interpolation. Hermite interpolation can deal with the derivative value, but using Hermite interpolation puts a restriction on the distribution of the share because not only $p'(x_1)$ but also $p(x_1)$ needs to be given as a share. Birkhoff interpolation can resolve that restriction.

3.2.1 Birkhoff Interpolation

Let $G = \{g_0, g_1, \cdots, g_N\}$ be a system of linearly independent, n times continuously differentiable real-valued functions in an interval $[a, b]$. A linear combination $P = \sum_{k=0}^{N} a_k g_k$ with real a_k will be called a polynomial in the system G. There exists an $m \times (n + 1)$ interpolation matrix

$$E = [e_{i,j}]_{i=1,j=0}^{m \quad n}, \quad m \geq 1, n \geq 0.$$

Its elements $e_{i,j}$ are zero or one and $\sum e_{i,j} = N + 1$ but no empty rows or namely an i for which $e_{i,j} = 0, j = 0, \cdots, n$. Then, let $X = \{x_1, \cdots, x_m\}$ be a given set of m distinct points in $[a, b]$, where $x_1 < \cdots < x_m$. The Birkhoff interpolation problem [36, 37] that corresponds to the triplet $\langle E, X, G\rangle$ and given data $c_{i,j}$ must find a polynomial p of degree at most n, that satisfies the conditions

$$p^{(j)}(x_i) = c_{i,j}, \quad e_{i,j} = 1. \tag{3.1}$$

The system (3.1) consists of $N + 1$ linear equations. The triplet $\langle E, X, G\rangle$ has a unique solution for each given set of $c_{i,j}$ if and only if the determinant of the system

$$D(E, X, G) = \det[g_0^{(j)}(x_i), \cdots, g_N^{(j)}(x_i); e_{i,j} = 1] \tag{3.2}$$

is different from zero. Formula (3.2) displays only one row of the determinant or namely the row corresponding to a pair (i, j) with $e_{i,j} = 1$. We denote the $(N + 1) \times (N + 1)$ matrix that appears in (3.2) as $A(E, X, G)$, the determinant is equal to $D(E, X, G) = |A(E, X, G)|$. When $c_{i,j} = p^{(j)}(x_i)$ are given, the interpolation polynomial is given by

$$p(x) = \sum_{j=0}^{N} \frac{D(E, X, G_j)}{D(E, X, G)} \cdot g_j(x), \tag{3.3}$$

where G_j is the set of functions obtained from G by replacing g_j with p, e.g., $G_1 = \{g_0, p, g_2, \cdots, g_N\}$.

3.2.2 Brief Example of Birkhoff Interpolation

$g_0(x) = 1, g_1(x) = x, g_2(x) = x^2$, that is, $G = \{1, x, x^2\}$ are given. X and E are also given as follows:

$$X = \{1, 2, 3\}, \quad E = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

When the data $c_{i,j}$ are actually given as $p(1) = 15$, $p(2) = 29$, $p'(3) = 23$, we look for a polynomial $p(x) = \sum_{j=0}^{2} a_j x^j$ satisfying $p(1) = c_{1,0} = 15$, $p(2) = c_{2,0} = 29$, $p'(3) = c_{3,1} = 23$.

$$D(E, X, G) = \begin{vmatrix} g_0(x_1) & g_1(x_1) & g_2(x_1) \\ g_0(x_2) & g_1(x_2) & g_2(x_2) \\ g_0'(x_3) & g_1'(x_3) & g_2'(x_3) \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 0 & 1 & 6 \end{vmatrix} = 3,$$

$$D(E,X,G_0)=\begin{vmatrix} p(x_1) & g_1(x_1) & g_2(x_1) \\ p(x_2) & g_1(x_2) & g_2(x_2) \\ p'(x_3) & g_1'(x_3) & g_2'(x_3) \end{vmatrix} = \begin{vmatrix} 15 & 1 & 1 \\ 29 & 2 & 4 \\ 23 & 1 & 6 \end{vmatrix} = 21,$$

$$D(E,X,G_1)=\begin{vmatrix} g_0(x_1) & p(x_1) & g_2(x_1) \\ g_0(x_2) & p(x_2) & g_2(x_2) \\ g_0'(x_3) & p'(x_3) & g_2'(x_3) \end{vmatrix} = \begin{vmatrix} 1 & 15 & 1 \\ 1 & 29 & 4 \\ 0 & 23 & 6 \end{vmatrix} = 15,$$

$$D(E,X,G_2)=\begin{vmatrix} g_0(x_1) & g_1(x_1) & p(x_1) \\ g_0(x_2) & g_1(x_2) & p(x_2) \\ g_0'(x_3) & g_1'(x_3) & p'(x_3) \end{vmatrix} = \begin{vmatrix} 1 & 1 & 15 \\ 1 & 2 & 29 \\ 0 & 1 & 23 \end{vmatrix} = 9,$$

$$p(x)=\sum_{j=0}^{2}\frac{D(E,X,G_j)}{D(E,X,G)}\cdot g_j(x)=7+5x+3x^2.$$

We consider a HSSS over prime field GF(p) where the prime number is sufficiently large. For example, the shares are given by $p(x)=3x^2+5x+7 \bmod p$ and $f'(x)=6x+5 \bmod p$, and also $p(1)=15, p(2)=29, p'(3)=23$ are available for recovering the secret, we obtain secret s with

$$s=p(0)=\frac{D(E,X,G_0)}{D(E,X,G)}=\frac{21}{3}=7.$$

3.3 An Issue with Tassa's Scheme for Finite Fields of Characteristic Two

As stated in [52], when we differentiate a polynomial $f(x)$ over finite fields of characteristic two, every term that is an even degree will disappear since differentiating x^i over the extension field such that i is an even number results in zero. Given $p(x)=a_2x^2+a_1x+a_0 \in \mathrm{GF}(2^L)[x]$, where $a_2,a_1,a_0 \in \mathrm{GF}(2^L)$, as an example, we obtain $p'(x)=a_1$ and $p''(x)=0$. For that reason, Tassa's scheme does not work as expected over finite fields of characteristic two. Therefore, in order to realize a $(\{1,k\},n)$ HSSS, we choose a polynomial $p(x) \in \mathrm{GF}(2^L)[x]$ that consists of the free coefficient and $k-1$ terms, each of which is an odd degree with random coefficients, that is,

$$p(x)=\sum_{i=1}^{k-1}a_ix^{2(i-1)+1}+s \in \mathrm{GF}(2^L)[x].$$

However, when we consider the more general scheme of a $(\{k_0, k_1\}, n)$ HSSS with $2 \leq k_0 < k_1$, we cannot realize such a HSSS effectively because the k_0-th derivative is needed and the k_0-th derivative of $p(x)$ always results in $p^{(k_0)}(x) = 0$.

3.4 Proposed Scheme

We describe the proposed $(\mathbf{k}, n)$ HSSS over finite fields of characteristic two that satisfies Definition 1.

3.4.1 Generalization

We again consider the issue shown in Section 3.3, or namely, a $(\{k_0, k_1\}, n)$ HSSS with $k_0 \geq 2$. The issue is that the k_0-th derivative of $p(x)$ always results in $p^{(k_0)}(x) = 0$. Therefore, what we need to look at is whether we can give a polynomial such that meaningful shares can be generated in the k_0-th derivative where $k_0 \geq 2$, and we reconsider the need to use derivatives. In order to realize the hierarchy, it is important that the constant term of the polynomial disappears every time we differentiate it, but we begin to realize that there is no need to use the definition of the derivative itself as follows:

$$f^{(n)}(x) = \begin{cases} \displaystyle\sum_{i=0}^{k-1-n} {}_{i+n}\mathrm{P}_n \cdot a_{i+n}x^i & (k-1 \geq n) \\ 0 & (k-1 < n) \end{cases},$$

$${}_n\mathrm{P}_r = \frac{n!}{(n-r)!} = n(n-1)\cdots(n-r+1),$$

where here $n \geq 0$ and $k = k_m$. Therefore, we do not use the n-th derivative $f^{(n)}(x) \in \mathrm{GF}(2^L)[x]$ of the polynomial $f(x)$ of degree $k-1$, but we instead define a function $f^{[n]}(x) \in \mathrm{GF}(2^L)[x]$ that is used to reduce each exponent of the variable x in the polynomial $f(x)$ n times, and the function $f^{[n]}(x)$ is hereinafter referred to as the n-th order reduction of $f(x)$, i.e.,

$$f^{[n]}(x) = \begin{cases} \displaystyle\sum_{i=0}^{k-1-n} a_{i+n}x^i & (k-1 \geq n) \\ 0 & (k-1 < n) \end{cases}, \tag{3.4}$$

where here $n \geq 0$ and $k = k_m$. This function is used to realize the hierarchy. For example, given $f(x) = a_2x^2 + a_1x + a_0$, we obtain $f^{[1]}(x) = a_2x + a_1$ and $f^{[2]}(x) = a_2$.

3.4.2 Birkhoff Interpolation Using the n-th Order Reduction

We consider whether Birkhoff interpolation can be applied to the HSSS using the n-th order reduction $f^{[n]}(x)$ to recover the secret. If Formula (3.3) holds even when $f^{[n]}(x)$ is used, we may apply to it Birkhoff interpolation to recover the secret. We then introduce Theorem 1 from Lemmas 1 and 2.

Lemma 1. *Suppose $F = [a(i,j)]$ is an $n \times n$ matrix and fix any $i, j \in \{1, 2, \cdots, n\}$. The following equation then holds for any $1 \leq i \leq n$, $1 \leq k \leq n$.*

$$\sum_{j=1}^{n} (-1)^{i+j} |\tilde{F}(i,j)| a(k,j) = \begin{cases} |F| & (k = i) \\ 0 & (k \neq i), \end{cases} \tag{3.5}$$

where $\tilde{F}(i,j)$ is the $(n-1) \times (n-1)$ submatrix of F formed by deleting the i-th row and the j-th column.

Proof. When an i is chosen from any $1 \leq i \leq n$, we prove that Eq. (3.5) holds for $k = i$ and $k \neq i$, respectively.

For the case of $k = i$, the left-hand side is the value of the cofactor expansion, also called the Laplace expansion, of $|F|$ along the i-th row. Thus, the equation holds for $k = i$.

For the case of $k \neq i$, when we do not care about the sign of the whole expression of the left-hand side itself, it is the value of the cofactor expansion along the i-th row for the determinant of $n \times n$ matrix

$$F = \begin{bmatrix} a(1,1) & \dots & a(1,n) \\ \vdots & & \vdots \\ a(k,1) & \dots & a(k,n) \\ \vdots & & \vdots \\ a(n,1) & \dots & a(n,n) \end{bmatrix} \begin{matrix} \text{the 1st row} \\ \vdots \\ \text{the } i\text{-th row} \\ \vdots \\ \text{the } n\text{-th row} \end{matrix},$$

where the i-th row is $[a(k,1), \cdots, a(k,n)]$ and the $l(\neq i)$-th row is $[a(l,1), \cdots, a(l,n)]$. However, the k-th row is also $[a(k,1), \cdots, a(k,n)]$. Thus, the equation holds for $k \neq i$ since whenever two rows of a matrix are identical, its determinant is zero. □

Lemma 2. *Suppose $F = [a(i,j)]$ is an $n \times n$ matrix and $FQ(j)$ is an $n \times n$ matrix where the j-th column of the matrix F is replaced with $[q(1), \cdots, q(n)]^{\mathrm{T}}$, that is,*

$$FQ(j) = \begin{bmatrix} a(1,1) & \dots & q(1) & \dots & a(1,n) \\ \vdots & & \vdots & & \vdots \\ a(n,1) & \dots & q(n) & \dots & a(n,n) \end{bmatrix}.$$

Then the following equation holds for any $1 \leq k \leq n$.

$$\sum_{j=1}^{n} |FQ(j)| a(k,j) = q(k) \times |F|. \tag{3.6}$$

Proof. The cofactor expansion along the j-th column for $|FQ(j)|$ yields

$$\sum_{i=1}^{n} (-1)^{i+j} |\tilde{F}(i,j)| q(i).$$

The left-hand side of Eq. (3.6) is expanded by using Lemma 1 as follows:

$$\begin{aligned} \sum_{j=1}^{n} |FQ(j)| a(k,j) &= \sum_{j=1}^{n} \sum_{i=1}^{n} (-1)^{i+j} |\tilde{F}(i,j)| q(i) a(k,j) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} (-1)^{i+j} |\tilde{F}(i,j)| q(i) a(k,j) \\ &= \sum_{i=1}^{n} q(i) \sum_{j=1}^{n} (-1)^{i+j} |\tilde{F}(i,j)| a(k,j) \\ &= q(k) \times |F| \end{aligned}$$

The proof is therefore complete. □

Theorem 1. *Assume that $D(E, X, G) \neq 0$. Formula (3.3), Birkhoff interpolation, holds even when the n-th order reduction $f^{[n]}(x)$ is used instead of the n-th derivative $f^{(n)}(x)$.*

Proof. The equation that needs to be satisfied is

$$p(x) \cdot D(E, X, G) = \sum_{j=0}^{N} D(E, X, G_j) \cdot g_j(x). \tag{3.7}$$

Note that Eq. (3.7) is equivalent to

$$p^{[j]}(x) \cdot D(E, X, G) = \sum_{j=0}^{N} D(E, X, G_j) \cdot g_j^{[j]}(x)$$

because $p^{[j]}(x)$ is uniquely determined by the polynomial of $p(x)$ under the n-th order reduction.

In the $(\mathbf{k}, n)$ HSSS with $\mathbf{k} = \{k_i\}_{i=0}^{m}$, Eq. (3.7) must hold for $x = x_1, \cdots, x_{k_m}$, where $N = k_m - 1$. Without loss of generality, we may assume that the 1st, $\cdots, k_0$-th participants in the highest level $\mathcal{U}_0$, the (k_0+1)-th, $\cdots, k_1$-th participants in the level $\mathcal{U}_1$, and participants up to the level $\mathcal{U}_m$ in the same manner are assigned. Then the relation between $q(i)$ of Lemma 2 and a share passed into Birkhoff interpolation can be represented as follows:

$$\begin{aligned}
&q(1) = p(x_1), \cdots, q(k_0) = p(x_{k_0}),\\
&q(k_0+1) = p^{[k_0]}(x_{k_0+1}), \cdots, q(k_1) = p^{[k_0]}(x_{k_1}),\\
&\quad\vdots\\
&q(k_{m-1}+1) = p^{[k_{m-1}]}(x_{k_{m-1}+1}), \cdots, q(k_m) = p^{[k_{m-1}]}(x_{k_m}).
\end{aligned}$$

Moreover, let F be a $k_m \times k_m$ matrix, where only the $j(= 1, \cdots, k_m)$-th column is shown, as follows:

$$F = \begin{bmatrix} a(1,j) \\ \vdots \\ a(k_0,j) \\ a(k_0+1,j) \\ \vdots \\ a(k_1,j) \\ \vdots \\ a(k_{m-1}+1,j) \\ \vdots \\ a(k_m,j) \end{bmatrix} = \begin{bmatrix} g_{j-1}(x_1) \\ \vdots \\ g_{j-1}(x_{k_0}) \\ g_{j-1}^{[k_0]}(x_{k_0+1}) \\ \vdots \\ g_{j-1}^{[k_0]}(x_{k_1}) \\ \vdots \\ g_{j-1}^{[k_{m-1}]}(x_{k_{m-1}+1}) \\ \vdots \\ g_{j-1}^{[k_{m-1}]}(x_{k_m}) \end{bmatrix}.$$

$D(E, X, G)$ equals the determinant of the matrix F and $D(E, X, G_j)$ equals $|FQ(j+1)|$, where $j = 0, \cdots, k_m - 1$. When we note that $k = 1, \cdots, k_m$ correspond to $x = x_1, \cdots, x_{k_m}$, respectively, we obtain

$$\begin{aligned}\sum_{j=0}^{N=k_m-1} D(E, X, G_j) \cdot g_j(x) &= \sum_{j=1}^{k_m} |FQ(j)| \cdot a(k, j) \\ &= q(k) \times |F| \\ &= p(x) \cdot D(E, X, G).\end{aligned}$$

The proof is thus complete since we have confirmed that Birkhoff interpolation works well by using Lemma 2. □

Theorem 1 means that there is no need for shares to be generated with the definition of the derivative itself and that the n-th order reduction $f^{[n]}(x)$ works in Birkhoff interpolation to recover the secret.

3.4.3 Distribution Algorithm

The dealer selects a random polynomial

$$p(x) = \sum_{i=0}^{k-1} a_i x^i \in \mathrm{GF}(2^L)[x], \quad a_0 = s.$$

The dealer identifies each participant $u \in \mathcal{U}$ with an element of $\mathrm{GF}(2^L)$. For simplicity, the element that corresponds to $u \in \mathcal{U}$ will be also denoted by u. The dealer securely distributes each of these shares to a corresponding participant in the following manner: Each participant $u \in \mathcal{U}_i$ receives the share $p^{[k_{i-1}]}(u)$, where $k_{-1} = 0$ and $p^{[n]}(x)$ is the definition (3.4).

Here, we describe the $(\{3, 4, 6\}, n)$ HSSS as an example, where $k_0 = 3, k_1 = 4$, and $k_2 = 6$. As $k = k_2 = 6$, the dealer selects a random polynomial of degree five, $p(x) = \sum_{i=1}^{5} a_i x^i + s \in \mathrm{GF}(2^L)[x]$. Then the dealer distributes the shares, that is, each participant $u \in \mathcal{U}_0$ will get the share $p(u)$, each participant $u \in \mathcal{U}_1$ will get the share $p^{[3]}(u) = \sum_{i=0}^{2} a_{i+3} u^i$, and each participant $u \in \mathcal{U}_2$ will get the share $p^{[4]}(u) = \sum_{i=0}^{1} a_{i+4} u^i$.

3.4.4 Recovery Algorithm

k participants cooperate to recover secret s. The secret is recovered from the shares, including the n-th order reduction $p^{[n]}(x)$, by using Theorem 1 and Birkhoff interpolation

as follows:

$$s = p(0) = \frac{D(E, X, G_0)}{D(E, X, G)}.$$

Here, we also describe the $(\{3, 4, 6\}, n)$ HSSS as an example, where $k_0 = 3, k_1 = 4$, and $k_2 = 6$. Table 3.1 shows six participants that includes three participants from $\mathcal{U}_0$ and five participants from $\mathcal{U}_0 \cup \mathcal{U}_1$ agree to recover the secret. Note that at least four participants from $\mathcal{U}_0 \cup \mathcal{U}_1$ are required to recover the secret. We obtain secret s with

Table 3.1: Participants and the shares for recovery

Participant	Share
$u_1 \in \mathcal{U}_0$	$p(u_1)$
$u_2 \in \mathcal{U}_0$	$p(u_2)$
$u_3 \in \mathcal{U}_0$	$p(u_3)$
$u_4 \in \mathcal{U}_1$	$p^{[3]}(u_4)$
$u_5 \in \mathcal{U}_1$	$p^{[3]}(u_5)$
$u_6 \in \mathcal{U}_2$	$p^{[4]}(u_6)$

the following $D(E, X, G)$ and $D(E, X, G_0)$, i.e.,

$$D(E, X, G) = \begin{vmatrix} 1 & u_1 & u_1^2 & u_1^3 & u_1^4 & u_1^5 \\ 1 & u_2 & u_2^2 & u_2^3 & u_2^4 & u_2^5 \\ 1 & u_3 & u_3^2 & u_3^3 & u_3^4 & u_3^5 \\ 0 & 0 & 0 & 1 & u_4 & u_4^2 \\ 0 & 0 & 0 & 1 & u_5 & u_5^2 \\ 0 & 0 & 0 & 0 & 1 & u_6 \end{vmatrix},$$

$$D(E, X, G_0) = \begin{vmatrix} p(u_1) & u_1 & u_1^2 & u_1^3 & u_1^4 & u_1^5 \\ p(u_2) & u_2 & u_2^2 & u_2^3 & u_2^4 & u_2^5 \\ p(u_3) & u_3 & u_3^2 & u_3^3 & u_3^4 & u_3^5 \\ p^{[3]}(u_4) & 0 & 0 & 1 & u_4 & u_4^2 \\ p^{[3]}(u_5) & 0 & 0 & 1 & u_5 & u_5^2 \\ p^{[4]}(u_6) & 0 & 0 & 0 & 1 & u_6 \end{vmatrix}.$$

3.4.5 Security Analysis

Theorem 2 proves the correctness and perfect privacy of our proposed $(\mathbf{k}, n)$ HSSS, where $\mathbf{k} = \{k_i\}_{i=0}^{m}$ and $k = k_m$. The proof is based on Tassa's approach in Section 3.1 of [9]. The main difference is that the n-th order reduction instead of the n-th order derivative is used in the coefficient matrix $M_{\mathcal{V}}$ to generate each participant's share that agrees to recover the secret.

Theorem 2. *Assume that the corresponding square matrix $M_{\mathcal{V}}$ including values of the n-th order reduction is regular,* $\det(M_{\mathcal{V}}) \neq 0$*, for any minimal authorized subset $\mathcal{V} \in \Gamma$, namely, $|\mathcal{V}| = k$. Then both correctness and perfect privacy hold.*

Proof. We first consider perfect privacy. A square matrix is regular, also called nonsingular, if and only if its determinant is nonzero. Equivalently, the rows of $M_{\mathcal{V}}$ are linearly independent. Let $\mathcal{V}_u \notin \Gamma$ be an unauthorized subset and $M_{\mathcal{V}_u}$ be the corresponding matrix. We aim at showing that even if all participants in $\mathcal{V}_u$ pool their shares together, they cannot reveal anything about secret s. This also implies that any value of s is accepted from their shares. The proof is that the secret is not included in the row space of $M_{\mathcal{V}_u}$, in the set of all possible linear combinations of the rows of $M_{\mathcal{V}_u}$. Without loss of generality, we may assume that $\mathcal{V}_u$ is missing only one participant to become authorized and we may boil the process down to adding to $\mathcal{V}_u$ the phantom participant $0 \in \mathcal{U}_0$ so that we can get an authorized subset. Then the square matrix corresponding to the authorized subset is regular by the assumption, the rows of the square matrix are linearly independent. As a result, the share of the participant $0 \in \mathcal{U}_0$ cannot be generated from $\mathcal{V}_u$, and the share is equivalent to the secret itself. In addition, even when $\mathcal{V}_u$ is missing only one participant in the j-th level, the access structure holds for adding one higher-level participant, that is, the highest-level participant $0 \in \mathcal{U}_0$.

Next, we consider correctness. Let $\mathcal{V} = \{v_1, \cdots, v_{|\mathcal{V}|}\} \subset \mathcal{U} = \bigcup_{i=0}^{m} \mathcal{U}_i$ and assume

that all the participants are assigned as follows:

$$\begin{aligned} &v_1, \cdots, v_{l_0} \in \mathcal{U}_0, \\ &v_{l_0+1}, \cdots, v_{l_1} \in \mathcal{U}_1, \\ &\quad \vdots \\ &v_{l_{m-1}+1}, \cdots, v_{l_m} \in \mathcal{U}_m, \end{aligned}$$

where $0 \leq l_0 \leq \cdots \leq l_m = |\mathcal{V}|$. $\mathcal{V}$ satisfies Definition 1 if and only if $l_i \geq k_i$ for all $0 \leq i \leq m$. The distribution of shares in Section 3.4.3 is represented as

$$\sigma(u) = \mathbf{r}^{(k_{i-1})}(u) \cdot \mathbf{a}$$

where the share $\sigma(u)$ of the participant $u \in \mathcal{U}_i$ and $\mathbf{a} = (s, a_1, \cdots, a_{k-1})^{\mathrm{T}}$ is the vector of coefficients of $p(x)$. Furthermore, let $\mathbf{r}(x) = (1, x, x^2, \cdots, x^{k-1})$ and let $\mathbf{r}^{(i)}(x)$ for all $i \geq 0$ denote the i-th order reduction (3.4) of that vector $\mathbf{r}(x)$. For example, $\mathbf{r}^{(1)}(x) = (0, 1, x, \cdots, x^{k-2})$. When all participants $v_1, \cdots, v_{l_m}$ of $\mathcal{V}$ pool together their shares of $\boldsymbol{\sigma} = (\sigma(v_1), \cdots, \sigma(v_{l_m}))^{\mathrm{T}}$, they need to solve $\boldsymbol{\sigma} = M_{\mathcal{V}} \cdot \mathbf{a}$ in the unknown vector $\mathbf{a}$. or in other words, secret s is obtained from

$$\begin{bmatrix} \sigma(v_1) \\ \vdots \\ \sigma(v_{l_m}) \end{bmatrix} = M_{\mathcal{V}} \begin{bmatrix} s \\ a_1 \\ \vdots \\ a_{k-1} \end{bmatrix}, M_{\mathcal{V}} = \begin{bmatrix} \mathbf{r}(v_1) \\ \vdots \\ \mathbf{r}(v_{l_0}) \\ \mathbf{r}^{(k_0)}(v_{l_0+1}) \\ \vdots \\ \mathbf{r}^{(k_0)}(v_{l_1}) \\ \cdots \\ \mathbf{r}^{(k_{m-1})}(v_{l_{m-1}+1}) \\ \vdots \\ \mathbf{r}^{(k_{m-1})}(v_{l_m}) \end{bmatrix}.$$

The proof is thus complete and $\det(M_{\mathcal{V}}) \neq 0$ is required for both correctness and perfect privacy. □

3.4.6 Unrecoverable Identities

The division by $D(E, X, G)$ is required for recovery. In other words, $D(E, X, G) \neq 0$ is required for the unique solution. Tassa described the probability in Section 3.2 of [9];

depending on the allocation of participant identities, there is a specific case in which the secret cannot be recovered in spite of the presence of an authorized subset because the determinant of a matrix used in recovery, related to the generator matrix, is zero under a certain probability. We looked into the probability of $D(E, X, G) = 0$ for the proposed scheme with experiments.

For recovery of three participants including two indispensable participants in a $(\{1, 3\}, n)$ HSSS over GF(2^8), we observed 10,795 patterns of $D(E, X, G) = 0$ in the combination of 2,731,135($= \binom{255}{3}$) patterns. This percentage is approximately $1/2^8$. Under the same observation in GF(2^{16}), the percentage is approximately $1/2^{16}$. As Tassa stated, the probability of $\det(M_{\mathcal{V}}) = 0$ is less than $1/(q-3)$ in a finite field of size q. Thus it is reasonable to think the same probability will be obtained for the proposed scheme and that there are almost no issues when a large finite field of characteristic two is used.

3.5 Software Implementation

For the $(\mathbf{k}, n)$ HSSS, we evaluated our scheme on the test environment, as described in Section 2.11. Each determinant of $D(E, X, G)$ and $D(E, X, G_0)$ was calculated with its triangular matrix since the determinant of a triangular matrix equals the product of the diagonal entries. Table 3.6 shows our experimental results.

With a $(\{1, 3\}, n)$ HSSS where the total number of participants is 60, we obtained a 97 Mbps recovery speed. However, that result shows 1/10th the speed of the optimization version specially made for a $(\{1, k\}, n)$ HSSS [52] over GF(2^8) with $k = 3$. The reason for the speed is mainly that the determinant calculation of a $k \times k$ matrix is generically implemented. Actually, we obtained a 970 Mbps recovery speed, shown later in Section 3.5.1, with the optimization version of the proposed scheme.

Table 3.2 shows the number of recovery operations for an input of size 888,710 bytes as an example for the optimized $(\{1, 3\}, n)$ HSSS over GF(2^8), shown later in Section 3.5.1. We see that the case in which the lookup table is not used (w/o table) increased the multiplication operation used in the division operation and both the additive operation and the shift operation used in the multiplication operation. Furthermore, Table 3.3 shows the number of the first $L = 8$-bit recovery operations, including the operations related to the identities that are processed only once.

Here, note that the optimization version of this scheme and that of [52] are different

Table 3.2: The number of recovery operations for an input of size 888,710 bytes

	One indispensable participant		Two indispensable participants	
	w/ table	w/o table	w/ table	w/o table
add	1,777,423	98,660,932	1,777,424	99,534,805
mul.	2,666,132	15,108,072	2,666,134	15,108,074
div.	888,710	888,710	888,710	888,710
shift	0	120,864,576	0	120,864,592
copy	2,666,133	26,661,305	2,666,134	26,661,308

Table 3.3: The number of the first 8-bit recovery operations

	One indispensable participant		Two indispensable participants	
	w/ table	w/o table	w/ table	w/o table
add	5	122	6	132
mul.	5	19	7	21
div.	1	1	1	1
shift	0	152	0	168
copy	6	35	7	38

implementations and that their speeds are the same. In [52], the dealer selects a random polynomial $p(x) = \sum_{i=1}^{k-1} a_i x^{2(i-1)+1} + s \in \mathrm{GF}(2^L)[x]$, where $a_i, s \in \mathrm{GF}(2^L)$, and we obtain $p'(x) = \sum_{i=1}^{k-1} a_i x^{2(i-1)} \in \mathrm{GF}(2^L)[x]$. The reason for the same speed is that the dominant computational cost of the optimization version and that of [52] are the same in a large file size of 888,710 bytes. More specifically, their operations related to the identities are different, but those are processed only once and are not dominant in a large file. In the optimization version of [52], the shares $p(u_1)$, $p'(u_2)$, $p'(u_3)$ instead of $p(u_1)$, $p^{[1]}(u_2)$, $p^{[1]}(u_3)$ shown in Table 3.4 are given and let $c_1 = p(u_1)$, $c_2 = p'(u_2)$,

$c_3 = p'(u_3)$. We obtain

$$D(E, X, G) = u_2^2 + u_3^2,$$
$$D(E, X, G_0) = \begin{vmatrix} c_1 & u_1 & u_1^3 \\ c_2 & 1 & u_2^2 \\ c_3 & 1 & u_3^2 \end{vmatrix} = c_1(u_2^2 + u_3^2) + c_2u_1(u_1^2 + u_3^2) + c_3u_1(u_1^2 + u_2^2).$$

$D(E, X, G)$ can be calculated with $c_1 = 1$, $c_2 = 0$, and $c_3 = 0$ of $D(E, X, G_0)$. Furthermore, $D(E, X, G)$, $u_2^2 + u_3^2$, $u_1(u_1^2 + u_2^2)$, and $u_1(u_1^2 + u_3^2)$ were calculated only once each. Therefore, the dominant computational cost was required in the calculation regarding c_1, c_2, and c_3.

3.5.1 A $(\{1, 3\}, n)$ HSSS and Its Optimization

Taking practical use into consideration, we cover the optimization specialized for a $(\{1, 3\}, n)$ HSSS and describe the recovery by three participants including one indispensable participant and three participants including two indispensable participants.

Three Participants Including One Indispensable Participant

The shares shown in Table 3.4 are given and let $c_1 = p(u_1)$, $c_2 = p^{[1]}(u_2)$, $c_3 = p^{[1]}(u_3)$. Since $D(E, X, G)$ is calculated with $c_1 = 1$,$c_2 = 0$, and $c_3 = 0$ of $D(E, X, G_0)$, we obtain

$$D(E, X, G) = u_2 + u_3,$$
$$D(E, X, G_0) = \begin{vmatrix} c_1 & u_1 & u_1^2 \\ c_2 & 1 & u_2 \\ c_3 & 1 & u_3 \end{vmatrix} = c_1(u_2 + u_3) + c_2u_1(u_1 + u_3) + c_3u_1(u_1 + u_2).$$

When the size of the secret exceeds L bits in GF(2^L), the secret is divided into L-bit

Table 3.4: Three participants including one indispensable participant

Participant	Share
$u_1 \in \mathcal{U}_0$	$p(u_1)$
$u_2 \in \mathcal{U}_1$	$p^{[1]}(u_2)$
$u_3 \in \mathcal{U}_1$	$p^{[1]}(u_3)$

pieces and each share is generated from each of the pieces with each random polynomial, the random values can then be deleted after the shares are distributed. In General, the size of the secret, such as a file size of 1MB, exceeds L bits, but the recovery procedures will be repeated in the combination of u_1, u_2, u_3. Therefore, it is sufficient that the additive operations and the multiplications of u_1, u_2, u_3 are performed only once for the file, and the dominant computational cost is the calculation that requires c_1, c_2, and c_3. More specifically, it is sufficient that $D(E, X, G)$, $u_2 + u_3$, $u_1(u_1 + u_2)$, and $u_1(u_1 + u_3)$ are calculated only once, respectively. Thus, this recovery requires three multiplications in $c_1(u_2+u_3)$, $c_2u_1(u_1+u_3)$, and $c_3u_1(u_1+u_2)$, two additive operations in $D(E, X, G_0)$, and then one division operation in the calculation to derive the secret.

Three Participants Including Two Indispensable Participants

The shares shown in Table 3.5 are given and let $c_1 = p(u_1)$, $c_2 = p(u_2)$, $c_3 = p^{[1]}(u_3)$. Since $D(E, X, G)$ is calculated with $c_1 = 1$,$c_2 = 1$,$c_3 = 0$ of $D(E, X, G_0)$, we obtain

$$D(E, X, G) = u_2(u_2 + u_3) + u_1(u_1 + u_3) = (u_1 + u_2)(u_1 + u_2 + u_3),$$

$$D(E, X, G_0) = \begin{vmatrix} c_1 & u_1 & u_1^2 \\ c_2 & u_2 & u_2^2 \\ c_3 & 1 & u_3 \end{vmatrix} = c_1u_2(u_2 + u_3) + c_2u_1(u_1 + u_3) + c_3u_1u_2(u_1 + u_2).$$

Given the dominant computational cost in the same manner, this recovery requires

Table 3.5: Three participants including two indispensable participants

Participant	Share
$u_1 \in \mathcal{U}_0$	$p(u_1)$
$u_2 \in \mathcal{U}_0$	$p(u_2)$
$u_3 \in \mathcal{U}_1$	$p^{[1]}(u_3)$

three multiplications, two additive operations, and one division operation.

Identities to Always Recover the Secret

Each participant has a different identity. For the case of one indispensable participant, $D(E, X, G)$ is always nonzero, and the secret can be recovered by any combination of

the participants in an authorized subset. For the case of two indispensable participants, $D(E, X, G) \neq 0$ if $u_1 + u_2 + u_3 \neq 0$. Therefore, if every participant has an identity such that, for example, its least significant bit is always one, the secret can be recovered by any combination of the participants in an authorized subset, and our scheme can be operated over GF(2^8) up to 128 identities. Moreover, if every participant has some sequence of characters or some string corresponding to their identification, a 225-bit identity can be generated from a SHA-224 hash value of the string and the least significant bit of one. In doing so, our scheme can be operated over GF(2^{256}).

3.6 Comparison with Other Schemes

We compare other HSSSs with our optimization version over GF(2^8).

3.6.1 Tassa's Scheme

We consider the fastest implementation over GF(257), where the closest prime field to 2^8. Each of the multiplication and division operations used the lookup table in the same manner as our scheme. The computational cost was equivalently low, but the memory consumption was higher because each lookup table required an array of 16-bit values stored from 0 to 256. Each of the additive and subtractive operations also used the lookup table. The computational cost was close to our scheme using only XOR operations, but the memory consumption was much higher because our scheme required no lookup tables.

To achieve the *ideal* HSSS, we must generate shares for s_i, $0 \leq i \leq n$ where the secret $s = \sum_{i=0}^{n} s_i \times 257^i$. That required complicated operations in distribution and recovery such that the multiple-precision arithmetic is required and such that every bit size of shares equals the bit size of the secret. Our scheme required simple 8-bit shares. Therefore, our scheme has a theoretical advantage in implementation, performance, and memory consumption.

Depending on the test conditions such as the used instruction set and CPU cache, but we actually reached only 0.15 Mbps under the same test environment, shown in Section 2.11, because the computational cost of the multiple-precision arithmetic was high, while the GNU multiple-precision arithmetic library (GMP) was used under GMPbench 0.2 score: 2813.9.

3.6.2 Selçuk et al.'s Scheme

When the truncated version is applied to our recovery procedure of three participants including one indispensable participant, shown in Chapter 3.5.1, we obtain

$$\begin{aligned} D(E, X, G) &= u_2 u_3 (u_2 + u_3), \\ D(E, X, G_0) &= \begin{vmatrix} c_1 & u_1 & u_1^2 \\ c_2 & u_2 & u_2^2 \\ c_3 & u_3 & u_3^2 \end{vmatrix} \\ &= c_1 u_2 u_3 (u_2 + u_3) + c_2 u_1 u_3 (u_1 + u_3) + c_3 u_1 u_2 (u_1 + u_2). \end{aligned}$$

As for the computational cost, our scheme has a very small advantage, while there is no difference between the two functions for the dominant computational cost. The closer to L bits in GF(2^L) the file size is, the more advantageous our scheme is, due to the smaller operations related to the identities. Considering a $(\{1, 2\}, n)$ HSSS, all shares of non-indispensable participants are not the same for the truncated version since a multiplication with the participant identity takes place, while all shares of non-indispensable participants are the same for our function.

3.7 Conclusions of This Chapter

In today's computer architecture, operations over GF(2^L) is faster than operations over GF(p). We considered applying Tassa's approach to operations over GF(2^L) and then proposed a $(\mathbf{k}, n)$ HSSS over finite fields of characteristic two, applicable to any level of $\mathbf{k}$. In order to accomplish it, we introduced the n-th order reduction $f^{[n]}(x)$ and confirmed it worked in Birkhoff interpolation. Our scheme is also both *perfect* and *ideal*. Given the implementation used in our experiments applicable to any level, that is, not limited to the use of a specific level, we found that 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 97 Mbps. Moreover, given the implementation optimized for a $(\{1, 3\}, n)$ HSSS, our implementation on the same PC was able to recover the given secret at a processing speed of approximately 970 Mbps.

Table 3.6: Experiment results

Level **k**	Shares and speed for recovery
{1,3}	$p(7), p^{[1]}(14), p^{[1]}(17)$ 100.92 Mbps
	$p(2), p(3), p^{[1]}(8)$ 97.07 Mbps
{2,4}	$p(6), p(7), p^{[2]}(14), p^{[2]}(17)$ 63.19 Mbps
	$p(1), p(2), p(3), p^{[2]}(8)$ 59.79 Mbps
{2,3,5}	$p(6), p(7), p^{[2]}(14), p^{[3]}(24), p^{[3]}(27)$ 35.40 Mbps
	$p(1), p(2), p(3), p^{[2]}(8), p^{[3]}(27)$ 34.84 Mbps
{2,4,6,10}	$p(6), p(7), p^{[2]}(14), p^{[2]}(17), p^{[4]}(24),$ $p^{[4]}(27), p^{[6]}(34), p^{[6]}(35), p^{[6]}(37), p^{[6]}(39)$ 7.84 Mbps
	$p(1), p(2), p(3), p^{[2]}(8), p^{[2]}(9),$ $p^{[4]}(24), p^{[4]}(27), p^{[6]}(34), p^{[6]}(37), p^{[6]}(39)$ 7.57 Mbps
{3,7,11,14,17}	$p(5), p(6), p(7), p^{[3]}(14), p^{[3]}(15), p^{[3]}(17), p^{[3]}(19),$ $p^{[7]}(24), p^{[7]}(25), p^{[7]}(27), p^{[7]}(29), p^{[11]}(34),$ $p^{[11]}(37), p^{[11]}(39), p^{[14]}(44), p^{[14]}(47), p^{[14]}(49)$ 1.62 Mbps
	$p(1), p(2), p(3), p(5), p^{[3]}(8), p^{[3]}(9), p^{[3]}(14),$ $p^{[3]}(17), p^{[7]}(24), p^{[7]}(25), p^{[7]}(27), p^{[7]}(29),$ $p^{[11]}(34), p^{[11]}(37), p^{[11]}(39), p^{[14]}(44), p^{[14]}(47)$ 1.53 Mbps

As shown in Section 2.1, elements in GF(2^L) can be identified with polynomials $f(X) = \sum_{i=0}^{L-1} f_i X^i, f_i \in \mathrm{GF}(2)$. Identities are represented by decimal numbers of $f_{L-1} \cdots f_1 f_0$ binary.

CHAPTER 4

HSSS Based on Information Dispersal Techniques

In this chapter, we introduce our hierarchical IDA, which is a HSSS that can be applied to any level, published at [60] [2] . Our scheme is both *perfect* and *ideal*. We use operations with GF(2^L). For perfect privacy, an SSS depends on the fact that an adversary can only pool at most $k-1$ shares. Chen et al.'s (k, n) threshold scheme [18] also depends on this fact. 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 4.3.2. Our overall contributions are summarized as follows:

- We apply hierarchy to the generator matrix used in an IDA and realize a HSSS applicable to any level. We solve the aforementioned issues and provide mathematical proof.
- In a single hierarchy, or a non-hierarchical SSS, our scheme is more efficient in implementation than Chen et al.'s scheme [18] 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 GF(2^L). Then, this scheme is much faster than an approach by Tassa [8, 9].

[2] The concept of this scheme was published at [56].

4.1 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 [50], 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.

4.2 Chen et al.'s SSS

From [18], 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 [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.

4.2.1 Distribution Algorithm

Table 4.1 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 $w_x \in \{0,1\}^\lambda$. The details are shown in Section 4.2.3.

[3] 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 $\mathrm{GF}(2)$.

Table 4.1: Chen et al.'s distribution algorithm

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

4.2.2 Recovery Algorithm

Table 4.2 shows the corresponding recovery algorithm. The algorithm takes as input

Table 4.2: Chen et al.'s recovery algorithm

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

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 terms of $\mathbf{M}$, shown in Section 4.2.3, 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.

4.2.3 Perfect Privacy

To shed further light on their algorithm, shown in Table 4.1, we detail Steps 3 and 4 as follows. In Step 3, we illustrate $k \times n$ matrix

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

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 $j = 1, \cdots, k$. Furthermore, w_x concatenates the elements in the $x+1$-th column of $\mathbf{M}'$.

Matrix $\mathbf{M}$ is required to satisfy Requirement 1 below to achieve perfect privacy [18]. Since $\mathbf{M}'$ is constructed with cyclic shifts, it indeed satisfies the requirement.

Requirement 1. *Any set of at most $k-1$ participants to cooperate to recover the secret learns no information about at least one of the k elements from every column of $\mathbf{M}$.*

4.3 Proposed Scheme

In this section, we describe our proposed $(\mathbf{k}, n)$ HSSS 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.

4.3.1 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 $(\{2,3,5\}, n)$ HSSS 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 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 2.1. 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 [9] 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 this chapter compact, we do not discuss it further.

4.3.2 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 $d_1, d_2, d_3, a_1, \cdots, a_5$. Furthermore, let d_1, d_2 be used for $\mathcal{U}_0$ and let $d_3, a_1, \cdots, a_5$ be used for $\mathcal{U}_1$. If three elements in $\mathcal{U}_1$ are given, such as a_1, a_2, a_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, SSSs 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.

4.3.3 Distribution Algorithm

The dealer securely distributes each share $w_x \in \{0,1\}^\lambda$ to participant P_x. Table 4.3 shows this specific algorithm. The underlined portions show differences between the algorithm of Table 4.1 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 4.3: Our proposed distribution algorithm

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

4.3.4 Recovery Algorithm

Table 4.4 shows our recovery algorithm. This algorithm takes as input shares of par-

Table 4.4: Our proposed recovery algorithm

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

ticipants 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 4.2 and our algorithm. Since there are no cyclic shifts after $Share^{\mathrm{IDA}}$, all matrices $\mathbf{G}'$ used by $Recover^{\mathrm{IDA}}$ 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 $Recover^{\mathrm{IDA}}$. 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.

4.3.5 Security Analysis

Theorem 3 proves the correctness and perfect privacy of our 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 $Recover^{\mathrm{IDA}}$, 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 $Share^{\mathrm{IDA}}$ and obtain $\mathbf{R}_{j-1}$. Lemma 3 proves that $\mathbf{R}_{j-1}$ cannot reveal anything about the secret.

Lemma 3. *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 2.4.*

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 4.3 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.,

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

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

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

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 3. *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 3 is based on an approach by Tassa [8, 9]. 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. □

4.4 Software Implementation

For the $(\mathbf{k}, n)$ HSSS, we evaluated our scheme on the test environment, shown in Section 2.11. Table 4.5 shows our experimental results, with the number of participants represented as $(|\mathcal{U}_0|, \cdots, |\mathcal{U}_m|)$.

Table 4.5: Results of our experiments for recovery

Level $\mathbf{k}$	Number of participants	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

4.5 Conclusions of This Chapter

We applied a hierarchy to the generator matrix used in an IDA through repeated consideration. 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.

CHAPTER 5

XOR-based HSSS for a Small Number of Indispensable Participants

In this chapter, we propose a $(\{1,3\},n)$ XOR-based HSSS for a small number of indispensable participants, published at [54], specially made for three participants including one or two indispensable participants to recover the secret through Fujii et al.'s scheme [11]. Our scheme is both *perfect* and *ideal*.

5.1 Fujii et al.'s $(2,n)$ threshold SSS

From [11], their scheme is *ideal*. Let $\mathcal{P}=\{P_0,\cdots,P_{n-1}\}$ be a set of n participants. We equally divide the secret $s\in\{0,1\}^{d(n_p-1)}$ into n_p-1 blocks $s_1,s_2,\cdots,s_{n_p-1}\in\{0,1\}^d$, where n_p is a prime number such that $n_p\geq n$. s_0 denotes a d-bit zero sequence, that is, $s_0\in\{0\}^d$. $d>0$ denotes the bit size of every divided piece of the secret. Figure 5.1 shows the relation between the secret s and each divided piece of the secret. If n is a composite number, we generate n shares with the $(2,n_p)$ threshold SSS instead of the $(2,n)$ threshold SSS.

5.1.1 Distribution Algorithm

Table 5.1 shows the corresponding distribution algorithm. The index values are ele-

$$\underbrace{s_0}_{\{0\}^d}, \overbrace{s_1||s_2||\cdots||s_{n_p-2}||s_{n_p-1}}^{s=\{0,1\}^{d(n_p-1)}}$$

Figure 5.1: Relation between the secret and each divided piece of the secret

Table 5.1: Fujii et al.'s distribution algorithm

Input: $s \in \{0,1\}^{d(n_p-1)}$
Output: $(w_0, \cdots w_{n-1})$
1: $s_0 \leftarrow \{0\}^d$
2: $s_1 \| \cdots \| s_{n_p-1} \leftarrow s$
3: for $i \leftarrow 0$ to $n_p - 2$:
4: $\quad r_i \xleftarrow{\$} \{0,1\}^d$
5: for $i \leftarrow 0$ to $n-1$: /* not n_p */
6: $\quad$ for $j \leftarrow 0$ to $n_p - 2$:
7: $\quad\quad w_{(i,j)} \leftarrow s_{i-j} \oplus r_j$
8: $\quad w_i \leftarrow w_{(i,0)} \| \cdots \| w_{(i,n_p-2)}$
9: return $(w_0, \cdots, w_{n-1})$

Table 5.2: Structure of shares in (2, 5) threshold SSS

	$j=0$	$j=1$	$j=2$	$j=3$
w_0	r_0	$s_4 \oplus r_1$	$s_3 \oplus r_2$	$s_2 \oplus r_3$
w_1	$s_1 \oplus r_0$	r_1	$s_4 \oplus r_2$	$s_3 \oplus r_3$
w_2	$s_2 \oplus r_0$	$s_1 \oplus r_1$	r_2	$s_4 \oplus r_3$
w_3	$s_3 \oplus r_0$	$s_2 \oplus r_1$	$s_1 \oplus r_2$	r_3
w_4	$s_4 \oplus r_0$	$s_3 \oplus r_1$	$s_2 \oplus r_2$	$s_1 \oplus r_3$

ments of GF(n_p). At lines 3 and 4, the dealer chooses $n_p - 1$ pieces of d-bit random number $r_0, \cdots, r_{n_p-2}$ from $\{0,1\}^d$ independently from each other with uniform probability $1/2^d$. At lines 5 through 8, the dealer generates pieces of shares and constructs

shares w_i by concatenating those pieces. The dealer then securely distributes share w_i to participant P_i. Table 5.2 shows the structure of shares with $n = 5$.

Table 5.3: Fujii et al.'s recovery algorithm

Input: (w_{t_0}, w_{t_1})
Output: $s \in \{0,1\}^{d(n_p-1)}$

$s_0 \leftarrow \{0\}^d$
$t_2 \leftarrow t_1 - t_0$
for $(i \leftarrow 0; i < 2; i \leftarrow i + 1)$:
    $w_{(t_i,0)}|| \cdots ||w_{(t_i,n_p-2)} \leftarrow w_{t_i}$
if $t_0 < n_p - 1$:
    $i \leftarrow 0$
    while $t_0 - i \cdot t_2 \not\equiv n_p - 1 (\mathrm{mod} n_p)$:
        $s_{(i+1)\cdot t_2} \leftarrow w_{(t_0,t_0-i\cdot t_2)} \oplus w_{(t_1,t_0-i\cdot t_2)} \oplus s_{i\cdot t_2}$
        $i \leftarrow i + 1$
if $t_1 < n_p - 1$:
    $i \leftarrow 0$
    while $t_1 + i \cdot t_2 \not\equiv n_p - 1 (\mathrm{mod} n_p)$:
        $s_{-(i+1)\cdot t_2} \leftarrow w_{(t_0,t_1+i\cdot t_2)} \oplus w_{(t_1,t_1+i\cdot t_2)} \oplus s_{-i\cdot t_2}$
        $i \leftarrow i + 1$
return $s \leftarrow s_1|| \cdots ||s_{n_p-1}$

5.1.2 Recovery Algorithm

Table 5.3 shows the corresponding recovery algorithm. Let $P_{t_0}, P_{t_1} (t_0 < t_1)$ be participants who cooperate to recover the secret. Here, we consider that P_1 and P_3 from Table 5.2 cooperate to recover the secret as an example. Table 5.4 shows the shares. In Table 5.4, when we start the recovery from r_1 as a starting point, we obtain s_2 with r_1 and $s_2 \oplus r_1$. Next, when we start the recovery from r_3 as a starting point, we obtain s_3, r_0, s_1, r_2, s_4 in order. As a result, we obtain $s = s_1||s_2||s_3||s_4$.

Table 5.4: Recovery in (2, 5) threshold SSS

	$j=0$	$j=1$	$j=2$	$j=3$
w_1	$s_1 \oplus r_0$	r_1	$s_4 \oplus r_2$	$s_3 \oplus r_3$
w_3	$s_3 \oplus r_0$	$s_2 \oplus r_1$	$s_1 \oplus r_2$	r_3

5.2 Proposed Scheme

We propose an XOR-based $(\{1,3\},n)$ HSSS specially made for the access structure with limiting the upper number of participants in the highest level $\mathcal{U}_0$ as

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

where $k_0 = 1, k_1 = 3, \mathcal{U} = \bigcup_{i=0}^{1} \mathcal{U}_i, \mathcal{U}_0 \cap \mathcal{U}_1 = \emptyset$, and $1 \leq |\mathcal{U}_0| \leq 2$ from Definition 1.

The $(\{1,3\},n)$ HSSS uses an XOR-based $(2,n)$ threshold SSS. In other words, the secret $s \in \{0,1\}^{d(n_p-1)}$ needs to be equally divided into n_p-1 blocks $s_1, s_2, \cdots, s_{n_p-1} \in \{0,1\}^d$. s_0 denotes a d-bit zero sequence. $d > 0$ denotes the bit size of every divided piece of the secret. If n is a composite number, we generate n shares with the $(2,n_p)$ threshold SSS instead of the $(2,n)$ threshold SSS.

5.2.1 Distribution Algorithm

Let $\mathcal{P} = \{P_0, \cdots, P_{n-1}\}$ be a set of n participants, i.e., $|\mathcal{U}|$ participants and P_0, $\cdots$, $P_{|\mathcal{U}_0|-1}$ be the highest-level participants $u \in \mathcal{U}_0$. Also, let $|\mathcal{U}_0|$ be one or two. First, we pass a random number instead of the secret s into Fujii et al.'s $(2,n)$ threshold SSS to generate intermediate shares $w_0, \cdots, w_{n-1}$ of the participants $P_0, \cdots, P_{n-1}$, respectively. Next, in order to generate actual shares $W_0, \cdots, W_{n-1}$ for the secret, each of the intermediate shares $w_0, \cdots, w_{|\mathcal{U}_0|-1}$ of the participants $u \in \mathcal{U}_0$ is mixed with the secret. Concretely speaking, each share W_i of the participant $u \in \mathcal{U}_0$ is generated with the equation $W_i = w_i \oplus s$. The other intermediate shares of the participants remain as they are, that is, $W_i = w_i$. For instance, when $n = 5$ is given, we replace divided pieces $s_1, \cdots, s_4$ of the secret in Table 5.2 with divided pieces $R_1, \cdots, R_4$ of a random number as shown in Table 5.5 and generate the intermediate shares $w_0, \cdots, w_4$. Then,

the shares $W_0, \cdots, W_{|\mathcal{U}_0|-1}$ of the secret are generated with $W_i = w_i \oplus s$ and the other shares $W_{|\mathcal{U}_0|}, \cdots, W_4$ are identical to $w_{|\mathcal{U}_0|}, \cdots, w_4$, respectively.

Table 5.5: Structure of intermediate shares in $(\{1,3\},5)$ HSSS

	$j=0$	$j=1$	$j=2$	$j=3$
w_0	r_0	$R_4 \oplus r_1$	$R_3 \oplus r_2$	$R_2 \oplus r_3$
w_1	$R_1 \oplus r_0$	r_1	$R_4 \oplus r_2$	$R_3 \oplus r_3$
w_2	$R_2 \oplus r_0$	$R_1 \oplus r_1$	r_2	$R_4 \oplus r_3$
w_3	$R_3 \oplus r_0$	$R_2 \oplus r_1$	$R_1 \oplus r_2$	r_3
w_4	$R_4 \oplus r_0$	$R_3 \oplus r_1$	$R_2 \oplus r_2$	$R_1 \oplus r_3$

Table 5.6 denotes the distribution algorithm. The index values are elements of

Table 5.6: Distribution algorithm of $(\{1,3\},n)$ HSSS

Input: $s \in \{0,1\}^{d(n_p-1)}$

Output: $(w_0, \cdots, w_{|\mathcal{U}_0|-1})$ for $\mathcal{U}_0$, $(w_{|\mathcal{U}_0|}, \cdots, w_{n-1})$ for $\mathcal{U}_1$

1: $R_0 \leftarrow \{0\}^d$

2: $R_1 || \cdots || R_{n_p-1} \leftarrow R \leftarrow GEN(\{0,1\}^{d(n_p-1)})$

3: **for** $(i \leftarrow 0; i < n_p - 1; i \leftarrow i+1)$:

4: $\quad r_i \leftarrow GEN(\{0,1\}^d)$

5: **for** $(i \leftarrow 0; i < n; i \leftarrow i+1)$: /* not n_p */

6: $\quad$ **for** $(j \leftarrow 0; j < n_p - 1; j \leftarrow j+1)$:

7: $\quad\quad w_{(i,j)} \leftarrow R_{i-j} \oplus r_j$

8: $\quad w_i \leftarrow w_{(i,0)} || \cdots || w_{(i,n_p-2)}$

9: **for** $(i \leftarrow 0; i < |\mathcal{U}_0|; i \leftarrow i+1)$:

10: $\quad w_i \leftarrow w_i \oplus s$

11: **return** $(w_0, \cdots, w_{n-1})$

GF(n_p). The boldface line numbers show the differences in the algorithm as shown in Table 5.1. At line 2, the dealer chooses $n_p - 1$ pieces of d-bit random number $R_1, \cdots, R_{n_p-1}$. These pieces can be also prepared in the same manner of $r_0, \cdots, r_{n_p-2}$ at lines 3 and 4. At lines 5 through 8, the dealer makes pieces of intermediate shares w_i

with $R_1, \cdots, R_{n_p-1}$ and constructs w_i by concatenating those pieces. At lines 9 and 10, the dealer mixes the secret with each of the intermediate shares w_i for the participants $u \in \mathcal{U}_0$. Lastly, the dealer distributes each share W_i to the participant P_i since every w_i has already been identical at line 11 to W_i previously described.

5.2.2 Recovery Algorithm

Table 5.7 denotes the recovery algorithm. The index values are elements of GF(n_p). Let $P_{t_0}, P_{t_1} (t_0 < t_1)$ and P_{v_0} be participants who cooperate to recover the secret. Either $P_{t_0}, P_{t_1} \in \mathcal{U}_0, P_{v_0} \in \mathcal{U}_1$ or $P_{t_0}, P_{t_1} \in \mathcal{U}_1, P_{v_0} \in \mathcal{U}_0$ is able to recover the secret, and the same recovery algorithm is applied to each set of the participants to recover the secret. The boldface line numbers show the differences in the algorithm as shown in Table 5.3. At lines 9 and 15, we obtain $n_p - 1$ pieces of the random number $R_1, \cdots, R_{n_p-1}$ used for generating the intermediate shares. It should be noted that $r_0, \cdots, r_{n_p-2}$ are saved from the calculation process of $R_1, \cdots, R_{n_p-1}$ just to reuse the process at lines 17 through 19, and that $r_0, \cdots, r_{n_p-2}$ are identical to those in the distribution algorithm when $P_{t_0}, P_{t_1} \in \mathcal{U}_1$ and $P_{v_0} \in \mathcal{U}_0$ recover the secret.

Here we give two examples of the recovery procedure for $n = 5$. We first consider that $P_0 \in \mathcal{U}_0$ and $P_1, P_3 \in \mathcal{U}_1$ cooperate to recover the secret when the shares are generated with $|\mathcal{U}_0| = 1$. Table 5.8 shows the shares. We start the recovery from W_1 and W_3 with the $(2, n)$ threshold SSS to obtain $R_1, \cdots, R_4$. Along with the calculation result, $r_0, \cdots, r_3$ are also obtained. As a result, we obtain $s_1, \cdots, s_4$ from W_0 to recover the secret by concatenating those pieces.

Next, we consider that $P_0, P_1 \in \mathcal{U}_0$ and $P_3 \in \mathcal{U}_1$ cooperate to recover the secret when the shares are generated with $|\mathcal{U}_0| = 2$. Table 5.9 shows the shares. We start the recovery from W_0 and W_1 with the $(2, n)$ threshold SSS to be able to obtain $R_1, \cdots, R_4$ because each $r_j \oplus s_{j+1}$, where $j = 0, \cdots, 3$, can be regarded as a random number for the $(2, n)$ threshold SSS. Along with the calculation result, $r_0 \oplus s_1, \cdots, r_3 \oplus s_4$ are also obtained. As a result, we obtain $s_1, \cdots, s_4$ from W_3 to recover the secret by concatenating those pieces.

When we distribute a large file such as data size of 1MB, it will be divided into $d(n_p - 1)$-bit pieces and each piece is considered as a secret. Then, the shares are generated from each secret. In addition, random numbers used for each share can be deleted after the distribution of the share.

Table 5.7: Recovery algorithm of $(\{1,3\},n)$ HSSS

Input: $\{(w_{t_0},w_{t_1}),(w_{v_0})\}\in\{(\mathcal{U}_0),(\mathcal{U}_1)\}$ or $\{(\mathcal{U}_1),(\mathcal{U}_0)\}$
Output: $s\in\{0,1\}^{d(n_p-1)}$

1: $R_0\leftarrow\{0\}^d$
2: $t_2\leftarrow t_1-t_0$
3: **for** $(i\leftarrow 0; i<2; i\leftarrow i+1)$:
4: $\quad w_{(t_i,0)}||\cdots||w_{(t_i,n_p-2)}\leftarrow w_{t_i}$
5: **if** $t_0<n_p-1$:
6: $\quad i\leftarrow 0$
7: $\quad$ **while** $t_0-i\cdot t_2\not\equiv n_p-1(\mathrm{mod}n_p)$:
8: $\quad\quad r_{(t_0-i\cdot t_2)}\leftarrow w_{(t_0,t_0-i\cdot t_2)}\oplus R_{i\cdot t_2}$
9: $\quad\quad R_{(i+1)\cdot t_2}\leftarrow w_{(t_1,t_0-i\cdot t_2)}\oplus r_{(t_0-i\cdot t_2)}$
10: $\quad\quad i\leftarrow i+1$
11: **if** $t_1<n_p-1$:
12: $\quad i\leftarrow 0$
13: $\quad$ **while** $t_1+i\cdot t_2\not\equiv n_p-1(\mathrm{mod}n_p)$:
14: $\quad\quad r_{(t_1+i\cdot t_2)}\leftarrow w_{(t_1,t_1+i\cdot t_2)}\oplus R_{-i\cdot t_2}$
15: $\quad\quad R_{-(i+1)\cdot t_2}\leftarrow w_{(t_0,t_1+i\cdot t_2)}\oplus r_{(t_1+i\cdot t_2)}$
16: $\quad\quad i\leftarrow i+1$
17: $w_{(v_0,0)}||\cdots||w_{(v_0,n_p-2)}\leftarrow w_{v_0}$
18: **for** $(i\leftarrow 0; i<n_p-1; i\leftarrow i+1)$:
19: $\quad s_i\leftarrow w_{(v_0,i)}\oplus r_i\oplus R_{v_0-i}$
20: **return** $s\leftarrow s_1||\cdots||s_{n_p-1}$

Table 5.8: Recovery with one indispensable participant

	$j=0$	$j=1$	$j=2$	$j=3$
W_0	$r_0\oplus s_1$	$R_4\oplus r_1\oplus s_2$	$R_3\oplus r_2\oplus s_3$	$R_2\oplus r_3\oplus s_4$
W_1	$R_1\oplus r_0$	r_1	$R_4\oplus r_2$	$R_3\oplus r_3$
W_3	$R_3\oplus r_0$	$R_2\oplus r_1$	$R_1\oplus r_2$	r_3

Table 5.9: Recovery with two indispensable participants

	$j=0$	$j=1$	$j=2$	$j=3$
W_0	$r_0 \oplus s_1$	$R_4 \oplus r_1 \oplus s_2$	$R_3 \oplus r_2 \oplus s_3$	$R_2 \oplus r_3 \oplus s_4$
W_1	$R_1 \oplus r_0 \oplus s_1$	$r_1 \oplus s_2$	$R_4 \oplus r_2 \oplus s_3$	$R_3 \oplus r_3 \oplus s_4$
W_3	$R_3 \oplus r_0$	$R_2 \oplus r_1$	$R_1 \oplus r_2$	r_3

5.2.3 Perfect Privacy

For the perfect privacy of our scheme, when the conceivable sets of the shares generated from the secret $s \in \{0,1\}^{d(n_p-1)}$ for every unauthorized set T are given, we evaluate the possibility of these shares given with another secret $s' \in \{0,1\}^{d(n_p-1)}$. Note that these shares are generated from the secret s and are already given, and that we evaluate the possibility of these shares given with another secret s'.

First, for every set of the shares given by the participants from $\mathcal{U}_1$, the shares have no information on the secret. Therefore, the same shares can be given with the secret s' if the random numbers used for s' are coincidentally identical to those used for s.

Next, for every set of the shares given by the participants from $\mathcal{U}_0$, each of the shares has $r_i \oplus s_{i+1}$, where $i = 0, \cdots, n_p - 2$. Therefore, the same shares can be given with the secret s' if $r'_i = r_i \oplus s_{i+1} \oplus s'_{i+1}$ instead of r_i is selected.

Last, for one share given by the participant from $\mathcal{U}_0$ and one share given by the participant from $\mathcal{U}_1$, the same shares can be given with the secret s' when we start to update r_i come from $r_i \oplus s_{i+1}$ without R_i from the piece of the share given by the participant from $\mathcal{U}_0$. For example, with regards to W_0 and W_1 in Table 5.8, we replace r_0 with $r'_0 = r_0 \oplus s_1 \oplus s'_1$, and replace R_1 with $R'_1 = R_1 \oplus r_0 \oplus r'_0$. Next, r_1 remains as it is, and we replace R_4 with $R'_4 = R_4 \oplus r_1 \oplus s_2 \oplus s'_2$. Then, we replace r_2 with $r'_2 = r_2 \oplus R_4 \oplus R'_4$ and replace R_3, r_3, and R_2 in the same manner.

Thus, we confirm that these shares for every unauthorized set T can be generated from not only s but also s'. Also, we are able to guarantee with certainty that the secret cannot be recovered without the shares of the indispensable participants since the recovery is attempted in an unauthorized set. In addition, our scheme is *ideal* since every bit size of shares equals the bit size of the secret.

5.2.4 Calculation Efficiency

For our distribution algorithm in Table 5.6, the step at lines 5 through 8 requires no bitwise XOR operations to make one divided piece of intermediate share $w_{(i,j)}$ constructed with R_0, and d bitwise XOR operations to make $w_{(i,j)}$ constructed without R_0. Thus, the step requires $(n_p - 2)d$ bitwise XOR operations to make each intermediate share of $w_0, \cdots, w_{n_p-2}$. Also, the step requires $(n_p - 1)d$ bitwise XOR operations to make w_{n_p-1}. Therefore, the average number of XOR operations to make one share is

$$\frac{(n_p-1)(n_p-2)+(n_p-1)}{n_p}d$$
$$= \frac{(n_p-1)^2}{n_p}d = \frac{(n_p-1)}{n_p}|s|.$$

Since the step at lines 5 through 8 makes n shares, it requires an average of $\frac{n_p-1}{n_p}n|s|$ bitwise XOR operations. The next step at lines 9 and 10 requires at most $2|s|$ bitwise XOR operations. Hence, our distribution algorithm requires an average of

$$\left(\frac{n_p-1}{n_p}n+2\right)|s|$$

bitwise XOR operations and an average of $(n+1)|s|$ bitwise XOR operations if $n = n_p$.

For our recovery algorithm in Table 5.7, the step at lines 5 through 16 requires $(2(n_p-1)-2)d$ bitwise XOR operations since no bitwise XOR operations are required with R_0 at each one of lines 8 and 14. the next step at lines 17 through 19 requires $(2(n_p-1)-1)d$ bitwise XOR operations since no bitwise XOR operations are required with R_0 once. Therefore, our recovery algorithm requires

$$(4n_p-7)d = (4(n_p-1)-3)d = \left(4-\frac{3}{n_p-1}\right)|s|$$

bitwise XOR operations.

We cannot simply compare calculation efficiency of our scheme with that of SSSs because of the comparison between the HSSS and the SSS, and we do not think the comparison with the $(2, n)$ threshold SSS or the $(3, n)$ threshold SSS is appropriate. However, Table 5.10 and Table 5.11 will be good references for that comparison. In addition, lines 9 and 10 in Table 5.6 and lines 17 through 19 in Table 5.7 will not be required in Fujii et al.'s $(2, n)$ threshold scheme.

Table 5.10: The number of XOR operations of $n_p = n$

	Distribution	Recovery
Fujii et al.'s $(2, n)$ threshold SSS	$(n-1)\|s\|$	$2\left(1 - \frac{1}{n-1}\right)\|s\|$
Kurihara et al.'s $(3, n)$ threshold SSS [12]	$(2n-1)\|s\|$	$\left(2n - \frac{1}{n-1}\right)\|s\|$
Our $(\{1,3\}, n)$ HSSS	$(n+1)\|s\|$	$\left(4 - \frac{3}{n-1}\right)\|s\|$

Table 5.11: The number of XOR operations of $n = 5$

	Distribution	Recovery
Fujii et al.'s $(2, n)$ threshold SSS	$4\|s\|$	$1.5\|s\|$
Kurihara et al.'s $(3, n)$ threshold SSS	$9\|s\|$	$9.75\|s\|$
Our $(\{1,3\}, n)$ HSSS	$6\|s\|$	$3.25\|s\|$

5.2.5 Open Issue

Table 5.12 shows the shares of three indispensable participants. We can process the recovery where $R_1, \cdots, R_4$ and $r_0 \oplus s_1, \cdots, r_3 \oplus s_4$ are obtained in the same manner that the secret is recovered with two indispensable participants, but we cannot obtain $s_1, \cdots, s_4$ from W_3.

Table 5.12: Recovery with three indispensable participants in $(\{1,3\}, 5)$ HSSS

	$j=0$	$j=1$	$j=2$	$j=3$
W_0	$r_0 \oplus s_1$	$R_4 \oplus r_1 \oplus s_2$	$R_3 \oplus r_2 \oplus s_3$	$R_2 \oplus r_3 \oplus s_4$
W_1	$R_1 \oplus r_0 \oplus s_1$	$r_1 \oplus s_2$	$R_4 \oplus r_2 \oplus s_3$	$R_3 \oplus r_3 \oplus s_4$
W_3	$R_3 \oplus r_0 \oplus s_1$	$R_2 \oplus r_1 \oplus s_2$	$R_1 \oplus r_2 \oplus s_3$	$r_3 \oplus s_4$

There are no issues on practical use in our scheme specialized for one or two indispensable participants in the highest level, however, taking more general scheme into consideration, we are able to say that removing the limitation on the upper number of

participants in the highest level $\mathcal{U}_0$ will be desirable.

5.3 Software Implementation

We evaluated our scheme in the test environment, as described in Section 2.11. Table 5.13 shows the number of the recovery operations with three participants including one indispensable participant and three participants including two indispensable participants, respectively. No significant difference in the recovery time from the number of the indispensable participants appears as shown in Table 5.13. Furthermore, Table 5.13 shows the results of $d = 64$ since we tried out four values of $d = 8, 16, 32, 64$ and have found $d = 64$ is the fastest and roughly twice as fast as $d = 32$. Also, it shows a median of the 19 experiments for each recovery.

Table 5.13: Time of recovery operations and throughput for an input of size 888,710 bytes

n	**One indispensable participant**		**Two indispensable participants**	
5	0.000791sec	8.37Gbps	0.000791sec	8.37Gbps
13	0.000865sec	7.65Gbps	0.000905sec	7.31Gbps
23	0.000897sec	7.38Gbps	0.000937sec	7.06Gbps
59	0.000865sec	7.65Gbps	0.000875sec	7.56Gbps
109	0.000904sec	7.32Gbps	0.000904sec	7.32Gbps

For the processing time of distribution, we found that it is linearly increasing with n. More specifically, it takes about $0.0003 \times n$ seconds, that is, 0.0015 seconds with $n = 5$ to generate shares of 888,710 bytes as Table 5.14 shows the experimental results for each distribution with $|\mathcal{U}_0| = 2$. In addition, the processing time to generate random numbers was not included in the result.

5.4 Conclusions of This Chapter

We proposed an XOR-based $(\{1, 3\}, n)$ HSSS specialized for three participants including one or two indispensable participants to recover the secret. We found out that it was essential that we pass a random number into Fujii et al.'s $(2, n)$ threshold SSS to generate

Table 5.14: Time of distribution operations for an input of size 888,710 bytes

n	Time (sec)
5	0.001512
13	0.003527
23	0.006874
59	0.017720
109	0.034060

intermediate shares, and that we mix the secret only for each of the intermediate shares of the indispensable participants. We found that our implementation was able to recover a given secret at a processing speed of approximately 7.0 Gbps.

CHAPTER 6

XOR-based HSSS

In this chapter, we propose an XOR-based *ideal* HSSS, which can be applied to any level, published at [57, 58]. Our scheme then requires a minimal number of higher-level participants to recover the secret. In addition, when k participants at the highest level $\mathcal{U}_0$ cooperate to recover the secret, the proposed scheme can yield the same result as Kurihara et al.'s (k, n) threshold SSS [13].

Here, Kurihara et al. [10] proposed an XOR-based conjunctive HSSS, which can be applied to any level. We show the three main differences between our proposed and their schemes. First, their technique to achieve hierarchy differs. Their scheme achieved hierarchy by generating shares of lower-level participants without the secret and some random values. Next, their scheme cannot yield the same result as Kurihara et al.'s (k, n) threshold SSS [13] when k participants at the highest level $\mathcal{U}_0$ cooperate to recover the secret. Last, our scheme has mathematical proof for perfect privacy. Therefore, our approach is different from their scheme and can have the advantage of actively using Kurihara et al.'s proof techniques [13].

Our overall contributions are summarized as follows:

- We apply the general concept of hierarchy to Kurihara et al.'s XOR-based SSS and realize a simple XOR-based HSSS, which can be applied to any level.
- We need to further discuss this, but our scheme could eliminate an issue typically associated with such hierarchical schemes [8, 9]; depending on the allocation of participant identities, there is a specific case in which the secret cannot be recovered in spite of an authorized subset.

6.1 Kurihara et al.'s Scheme

From Kurihara et al.'s scheme [13], we equally divide the secret $s \in \{0,1\}^{d(n_p-1)}$ into $n_p - 1$ blocks $s_1, \cdots, s_{n_p-1} \in \{0,1\}^d$. d is, for example, 8 and 64. The dealer securely distributes each share w_i for $i = 0, \cdots, n-1$ of a (k, n_p) threshold scheme to participant P_i. Therefore, we can construct a (k, n) threshold scheme by using a (k, n_p) threshold scheme even if n is a composite number.

Table 6.1 shows the distribution algorithm. Step 1 divides the secret equally into

Table 6.1: Kurihara et al.'s distribution algorithm

Input: $s \in \{0,1\}^{d(n_p-1)}$

Output: $(w_0, \cdots w_{n-1})$

1: $s_0 \leftarrow \{0\}^d, s_1 || \cdots || s_{n_p-1} \leftarrow s$

2: for $i \leftarrow 0$ to $k-2$:

  for $j \leftarrow 0$ to $n_p - 1$:

    $r_j^i \xleftarrow{\$} \{0,1\}^d$ (discard $r_{n_p-1}^0$)

3: for $i \leftarrow 0$ to $n-1$:

  for $j \leftarrow 0$ to $n_p - 2$:

    $w_{(i,j)} \leftarrow F_W(i,j)$

  $w_i \leftarrow w_{(i,0)} || \cdots || w_{(i,n_p-2)}$

4: return $(w_0, \cdots w_{n-1})$

$F_W(i,j)$

F1: $w_{(i,j)} \leftarrow \left(\bigoplus_{h=0}^{k-2} r_{h \cdot i+j}^h \right) \oplus s_{j-i}$

F2: return $w_{(i,j)}$

- $\mathbf{s}$ denotes the vector $(s_1, \cdots, s_{n_p-1})^{\mathrm{T}}$.
- $\mathbf{r}$ denotes the vector $(r_0^0, \cdots, r_{n_p-2}^0, r_0^1, \cdots, r_{n_p-1}^1, \cdots, r_0^{k-2}, \cdots, r_{n_p-1}^{k-2})^{\mathrm{T}}$.
- $\mathbf{e}$ denotes the $(kn_p - 2)$-dimensional vector $\begin{bmatrix} \mathbf{r} \\ \mathbf{s} \end{bmatrix}$.
- $\mathbf{v}_{(i,j)}$ denotes a $(kn_p - 2)$-dimensional binary vector such that $w_{(i,j)} = \mathbf{v}_{(i,j)} \cdot \mathbf{e}$.

$n_p - 1$ pieces of the d-bit sequence. Step 2 generates $(k-1)n_p - 1$ pieces of the d-bit random number. Here, we define the vectors $\mathbf{s}, \mathbf{r}$ and $\mathbf{e}$, as given in Table 6.1. Step 3 generates shares w_i by using the function $F_W(i,j)$. We can view Step F1 as

$w_{(i,j)} = \mathbf{v}_{(i,j)} \cdot \mathbf{e}$ with a uniquely given vector $\mathbf{v}_{(i,j)}$ corresponding to the (k, n_p) threshold scheme. For example, $\mathbf{v}_{(2,1)} = (0100\ 00010\ 0001)$ if $k = 3, n_p = 5$.

Table 6.2 shows the recovery algorithm. Participants P_i for $i = t_0, \cdots, t_{k-1}$ cooperate to recover the secret. Step 1 equally divides each share into d-bit pieces. Step 2 generates the $k(n_p - 1)$-dimensional vector $\mathbf{w}$. In Step 3, we obtain the matrix $\mathbf{M}$ by using the function $F_{MAT}()$. Step 4 recovers $s_1, \cdots, s_{n_p-1}$ by calculating $\mathbf{M} \cdot \mathbf{w}$. In Step 5, we obtain secret s by concatenating $s_1, \cdots, s_{n_p-1}$.

Table 6.3 shows the function $F_{MAT}()$. In Step F1, we obtain the vectors $\mathbf{v}_{(t_i,j)}$

Table 6.2: Kurihara et al.'s recovery algorithm

Input: $(w_{t_0}, \cdots, w_{t_{k-1}})$
Output: s
1: for $i \leftarrow 0$ to $k-1$:
$w_{(t_i,0)} \| \cdots \| w_{(t_i,n_p-2)} \leftarrow w_{t_i}$
2: $\mathbf{w} \leftarrow (w_{(t_0,0)}, \cdots, w_{(t_0,n_p-2)}, \cdots,$
$w_{(t_{k-1},0)}, \cdots, w_{(t_{k-1},n_p-2)})^{\mathrm{T}}$
3: $\mathbf{M} \leftarrow F_{MAT}(t_0, \cdots, t_{k-1})$
4: $(s_1, \cdots, s_{n_p-1})^{\mathrm{T}} \leftarrow \mathbf{M} \cdot \mathbf{w}$
5: $s \leftarrow s_1 \| \cdots \| s_{n_p-1}$
6: return s

such that $w_{(t_i,j)} = \mathbf{v}_{(t_i,j)} \cdot \mathbf{e}$. In Step F2, we obtain the matrix $\mathbf{G}$. In Step F3, the matrix $\left[\mathbf{G}\ \mathbf{I}_{k(n_p-1)}\right]$ is transformed to the row echelon form $\left[\bar{\mathbf{G}}\ \mathbf{J}\right]$. Matrices $\bar{\mathbf{G}}$ and $\mathbf{J}$ correspond to the matrices transformed from $\mathbf{G}$ and $\mathbf{I}_{k(n_p-1)}$, respectively. We then view the matrix $\left[\bar{\mathbf{G}}\ \mathbf{J}\right]$ as a matrix divided into block matrices $\boldsymbol{O}, \mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2, \mathbf{J}_0$, and $\mathbf{J}_1$, as given in Table 6.3. Step F4 transforms the matrix $[\mathbf{G}_0\ \mathbf{J}_0]$ to the matrix $\left[\mathbf{I}_{n_p-1}\ \mathbf{M}\right]$. Step 5 outputs the matrix $\mathbf{M}$.

We analyze the proof techniques [13] to describe the correctness and perfect privacy of our scheme and to actively use their lemmas.

Table 6.3: $F_{MAT}()$

$F_{MAT}(t_0, \cdots, t_{k-1})$

F1: for $i \leftarrow 0$ to $k-1$:
  for $j \leftarrow 0$ to $n_p - 2$:
    $\mathbf{v}_{(t_i,j)} \leftarrow w_{(t_i,j)} = \mathbf{v}_{(t_i,j)} \cdot \mathbf{e}$

F2: $\mathbf{G} \leftarrow (\mathbf{v}_{(t_0,0)}, \cdots, \mathbf{v}_{(t_{k-1},n_p-2)})^{\mathrm{T}}$

F3: $\left[\begin{array}{cc:c} \mathbf{G}_2 & \mathbf{G}_1 & \mathbf{J}_1 \\ \boldsymbol{O} & \mathbf{G}_0 & \mathbf{J}_0 \end{array}\right] = \left[\begin{array}{c:c} \bar{\mathbf{G}} & \mathbf{J} \end{array}\right] \leftarrow \left[\mathbf{G}\ \mathbf{I}_{k(n_p-1)}\right]$

F4: $\left[\begin{array}{cc:c} \mathbf{G}_2 & \mathbf{G}_1 & \mathbf{J}_1 \\ \boldsymbol{O} & \mathbf{I}_{n_p-1} & \mathbf{M} \end{array}\right] = \left[\begin{array}{c:c} \bar{\bar{\mathbf{G}}} & \bar{\bar{\mathbf{J}}} \end{array}\right] \leftarrow \left[\begin{array}{c:c} \bar{\mathbf{G}} & \mathbf{J} \end{array}\right]$

F5: return $\mathbf{M}$

- $\mathbf{w}$ denotes a $k(n_p - 1)$-dimensional vector.
- $\mathbf{M}$ denotes a $(n_p - 1) \times k(n_p - 1)$ binary matrix.
- $\mathbf{G}$ denotes a $k(n_p - 1) \times (kn_p - 2)$ binary matrix.

Definition 3. *Let $t_0, \cdots, t_{L-1}$ denote the indexes of L shares, which are arbitrary numbers such that $0 \leq t_i, t_j \leq n-1$ and $t_i \neq t_j$ if $i \neq j$. Let $s_0 = \{0\}^d$ denote a singular point. Furthermore, let $\mathbf{r}$ be the same as shown in Table 6.1 and let $\mathbf{s}' = (s_0, \cdots, s_{n_p-1})^{\mathrm{T}}$. Then, we define the $L(n_p - 1) \times (kn_p - 1)$ binary generator matrix $\mathbf{G}' = [\mathbf{U}\ \mathbf{V}]$ such that*

$$\begin{aligned} \mathbf{w} &= \mathbf{G}' \cdot \begin{bmatrix} \mathbf{r} \\ \mathbf{s}' \end{bmatrix} = \mathbf{U} \cdot \mathbf{r} \oplus \mathbf{V} \cdot \mathbf{s}' \\ &= (w_{(t_0,0)}, \cdots, w_{(t_0,n_p-2)}, \cdots, w_{(t_{L-1},0)}, \cdots, w_{(t_{L-1},n_p-2)})^{\mathrm{T}}. \end{aligned}$$

We may discuss correctness and perfect privacy with $\mathbf{s}'$ by using this singular point because $\mathbf{w}$ does not change. Furthermore, we may set as zero or one each element of the column of $\mathbf{G}'$ corresponding to $s_0 = \{0\}^d$. Therefore, we can represent the matrix

$\mathbf{V}$ with $(n_p - 1) \times n_p$ matrices $\mathbf{E}_{(j)}$.

Lemma 4 ([13] Lemma 1)**.** *Under Definition 3, the conditions*

$$\begin{aligned} &rank(\mathbf{G}') = rank(\mathbf{U}) = L(n_p - 1) && \text{if } 1 \le L \le k-1, \\ &rank(\mathbf{G}') = k(n_p - 1),\ rank(\mathbf{U}) = (k-1)(n_p - 1) && \text{if } L \ge k \end{aligned}$$

are satisfied when matrices $\mathbf{G}'$ and $\mathbf{E}_{(j)}$ are determined, i.e.,

$$\begin{aligned} \mathbf{G}' &= \left[\begin{array}{c:c} \mathbf{U} & \mathbf{V} \end{array}\right] \\ &= \left[\begin{array}{c:ccc:c} \mathbf{I}_{n_p-1} & \mathbf{E}_{(t_0)} & \cdots & \mathbf{E}_{((k-2)t_0)} & \mathbf{E}_{((n_p-1)t_0)} \\ \mathbf{I}_{n_p-1} & \mathbf{E}_{(t_1)} & \cdots & \mathbf{E}_{((k-2)t_1)} & \mathbf{E}_{((n_p-1)t_1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{I}_{n_p-1} & \mathbf{E}_{(t_{L-1})} & \cdots & \mathbf{E}_{((k-2)t_{L-1})} & \mathbf{E}_{((n_p-1)t_{L-1})} \end{array}\right], \\ \mathbf{E}_{(j)} &= \left[\begin{array}{c:c:c} & 0 & \\ \boldsymbol{O} & \vdots & \mathbf{I}_{n_p-j} \\ & 0 & \\ \hdashline & 0 & \\ \mathbf{I}_{j-1} & \vdots & \boldsymbol{O} \\ & 0 & \end{array}\right]. \end{aligned}$$

Here, we define matrix $\mathbf{E}^{(2)}_{(i,j)} = \mathbf{E}_{(i)} \oplus \mathbf{E}_{(j)}$. We then perform elementary row operations on matrix $\mathbf{G}'$ and obtain matrix $[\mathbf{U}'\ \mathbf{V}']$ represented by matrices

$$\begin{aligned} \mathbf{U}' &= \left[\begin{array}{c:ccc} \mathbf{I}_{n_p-1} & \mathbf{E}_{(t_0)} & \cdots & \mathbf{E}_{((k-2)t_0)} \\ \hdashline \boldsymbol{O} & \mathbf{E}^{(2)}_{(t_0,t_1)} & \cdots & \mathbf{E}^{(2)}_{((k-2)t_0,(k-2)t_1)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \mathbf{E}^{(2)}_{(t_0,t_{L-1})} & \cdots & \mathbf{E}^{(2)}_{((k-2)t_0,(k-2)t_{L-1})} \end{array}\right] = \left[\begin{array}{c:c} \mathbf{I}_{n_p-1} & * \\ \hdashline \boldsymbol{O} & \\ \vdots & \mathbf{U}'' \\ \boldsymbol{O} & \end{array}\right], \\ \mathbf{V}' &= \left[\begin{array}{c} \mathbf{E}_{((n_p-1)t_0)} \\ \hdashline \mathbf{E}^{(2)}_{((n_p-1)t_0,(n_p-1)t_1)} \\ \vdots \\ \mathbf{E}^{(2)}_{((n_p-1)t_0,(n_p-1)t_{L-1})} \end{array}\right] = \left[\begin{array}{c} \mathbf{E}_{((n_p-1)t_0)} \\ \hdashline \mathbf{V}'' \end{array}\right]. \end{aligned}$$

We also define matrix $\mathbf{D} = [\mathbf{U}'' \ \mathbf{V}'']$, and Lemma 4 satisfies the conditions

$$\begin{aligned} &rank(\mathbf{D}) = rank(\mathbf{U}'') = (L-1)(n_p - 1) && \text{if } 1 \le L \le k-1, \\ &rank(\mathbf{D}) = (k-1)(n_p-1),\ rank(\mathbf{U}'') = (k-2)(n_p-1) && \text{if } L \ge k. \end{aligned}$$

Kurihara et al. introduced several other techniques in their proof of Lemma 4. Specifically, they introduced Lemmas 5 and 6, and defined matrix $\mathbf{H}_{(i,j)}$ by adding one row to $\mathbf{E}^{(2)}_{(i,j)}$ with all rows of $\mathbf{E}^{(2)}_{(i,j)}$.

Lemma 5 ([13] Lemma 4)**.** *Suppose p is a prime number. Let $i, j, l \in \mathrm{GF}(p), i \neq j$, and let $\mathbf{i}_l$ be a p-dimensional unit vector such that*

$$\mathbf{i}_l = (\overset{0}{\check{0}}, \cdots, 0, \overset{l}{\check{1}}, 0, \cdots, \overset{p-1}{\check{0}}).$$

Furthermore, let $\mathcal{X}$ be a set of p-dimensional binary vectors defined by

$$\mathcal{X} = \{\mathbf{i}_{i+m} \oplus \mathbf{i}_{j+m} \mid 0 \le m \le p-2\}.$$

Then, all vectors in $\mathcal{X}$ are linearly independent.

Lemma 6 ([13] Lemma 6)**.** *Suppose $x_0, \cdots, x_{i-1}$ for $i \ge 2$ are random numbers selected from the finite set $\{0,1\}^h$ for $h > 0$ independently from each other with uniform probability $1/2^h$. Let $\mathcal{X}$ be the set defined by $\mathcal{X} = \{x_0, \cdots, x_{i-1}\}$. Then, $x_0, \cdots, x_{i-1}$ and all XOR-ed combinations of the elements in $\mathcal{X}$ are random numbers that are pairwise independent and distributed uniformly over $\{0,1\}^h$.*

Definition 4. *Let $i, j \in \mathrm{GF}(n_p), i \neq j$. We define the matrix $n_p \times n_p$ as follows:*

$$\mathbf{H}_{(i,j)} = \begin{bmatrix} \mathbf{E}^{(2)}_{(i,j)} \\ \bigoplus_{l=0}^{n_p-2} (\mathbf{i}_{i+l} \oplus \mathbf{i}_{j+l}) \end{bmatrix}.$$

Then, using Lemma 5 and $\mathbf{L}_i$, which denotes a circulant matrix based on the identity

matrix, $\mathbf{H}_{(i,j)}$ *is represented as*

$$\mathbf{H}_{(i,j)} = \begin{bmatrix} \mathbf{E}^{(2)}_{(i,j)} \\ \mathbf{i}_{i-1} \oplus \mathbf{i}_{j-1} \end{bmatrix} = \mathbf{L}_i \oplus \mathbf{L}_j, \quad \mathbf{L}_i = \begin{bmatrix} \mathbf{E}_{(i)} \\ \mathbf{i}_{i-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{O} & \mathbf{I}_{n_p-i} \\ \mathbf{I}_i & \boldsymbol{O} \end{bmatrix}.$$

Since $\mathrm{rank}(\mathbf{E}^{(2)}_{(i,j)}) = n_p - 1$ from Lemma 5, $\mathrm{rank}(\mathbf{H}_{(i,j)}) = n_p - 1$. Furthermore, we can represent matrix $\mathbf{D} = [\mathbf{U}''\ \mathbf{V}'']$ as matrix $\mathbf{M} = [\mathbf{P}\ \mathbf{Q}]$, where

$$\mathbf{P} = \begin{bmatrix} \mathbf{H}_{(t_0,t_1)} & \cdots & \mathbf{H}_{((k-2)t_0,(k-2)t_1)} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{(t_0,t_{L-1})} & \cdots & \mathbf{H}_{((k-2)t_0,(k-2)t_{L-1})} \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} \mathbf{H}_{((n_p-1)t_0,(n_p-1)t_1)} \\ \vdots \\ \mathbf{H}_{((n_p-1)t_0,(n_p-1)t_{L-1})} \end{bmatrix}.$$

As a result, $\mathrm{rank}(\mathbf{D}) = rank(\mathbf{M})$, $\mathrm{rank}(\mathbf{U}'') = rank(\mathbf{P})$, and $\mathrm{rank}(\mathbf{V}'') = rank(\mathbf{Q})$ are satisfied. Then, we can simply multiply $\mathbf{H}_{(i,j)}$ required in elementary row operations because $\mathbf{L}_i \cdot \mathbf{L}_j = \mathbf{L}_j \cdot \mathbf{L}_i = \mathbf{L}_{i+j}$ is satisfied.

6.2 Proposed Scheme

In this section, we describe the proposed $(\mathbf{k}, n)$ HSSS that satisfies Definition 1. Our scheme requires a minimal number of higher-level participants to recover the secret. From another perspective, not only k_0 participants at the highest level but also $k_0 + 1$ or more participants at the highest level can participate in the recovery. Therefore, we construct the scheme such that it can lead to the same result as Kurihara et al.'s (k, n) threshold scheme [13] when k participants at the highest level cooperate to recover the secret.

6.2.1 Distribution Algorithm

We equally divide secret $s \in \{0,1\}^{d(n_p-1)}$ into $n_p - 1$ blocks $s_1, \cdots, s_{n_p-1} \in \{0,1\}^d$. d is, for example, 8 and 64. The dealer securely distributes each share w_i for $i = 0, \cdots, n-1$ of a $(\mathbf{k}, n_p)$ hierarchical scheme to participant P_i. Therefore, we can construct a $(\mathbf{k}, n)$ hierarchical scheme with a $(\mathbf{k}, n_p)$ hierarchical scheme even if n is a composite number.

Table 6.4 shows the difference between the proposed and Kurihara et al.'s algorithms.

Table 6.4: $F_W(i,j)$ of the proposed scheme

$$R(x) := \left(\bigoplus_{h=0}^{x} r_{h\cdot i+j}^{h}\right) \oplus \begin{cases} \text{if } x = k-2 : 0 \\ \text{else :} \quad \bigoplus_{h=x+1}^{k-2} \bigoplus_{a=0}^{n_p-1} r_a^h \end{cases}$$

F1: $w_{(i,j)} \leftarrow \begin{cases} \text{if } l = 0 : R(k-2) \oplus s_{j-i} \\ \text{else :} \quad R(k-1-k_{l-1}) \oplus \left(\bigoplus_{a=1}^{n_p-1} s_a\right) \end{cases}$

F2: return $w_{(i,j)}$

- Participant $P_i \in \mathcal{U}_l$ receives $w_{(i,j)}$ and k_l is defined by Definition 1.

6.2.2 Recovery Algorithm

This difference leads to corresponding changes in $\mathbf{G}$, $\mathbf{G}'$, and $\mathbf{v}_{(i,j)}$, but our recovery algorithm is identical to that of Kurihara et al., as shown in Tables 6.2 and 6.3. We can then view Step 3 of Table 6.2 as a precomputation.

6.2.3 Achieving Hierarchy

This creative solution of $F_W(i,j)$ in the proposed algorithm achieves hierarchy and is nontrivial, while the difference appears to be minor. Tassa's scheme [8, 9] achieves hierarchy by using the properties of polynomial derivatives; shares of lower-level participants are not generated from the secret and some random values. Because Step F1 in Table 6.4 means $w_{(i,j)} = \mathbf{v}_{(i,j)} \cdot \mathbf{e}$, we have attempted to set all elements corresponding to the secret and some random values of $\mathbf{v}_{(i,j)}$ to zero for the lower-level participants, but this approach did not work. Through repeated consideration, we can achieve hierarchy when those elements are all ones and prove the correctness and perfect privacy of our scheme by actively using Kurihara et al.'s proof techniques.

6.2.4 Allocation Issue

Achieving hierarchy with the proposed XOR-based scheme could eliminate a traditional issue. As Tassa stated in [8, 9], depending on the allocation of participant identities, there is a specific case in which the secret cannot be recovered in spite of the presence of an authorized subset because the determinant of a matrix used in recovery, related to

the generator matrix, is zero under a certain probability. Our scheme does not require values of participant identities in the generator matrix. However, we unfortunately found that there is a specific case in which the secret cannot be recovered in spite of the presence of an authorized subset, for example, as shown in Table 6.5. Therefore,

Table 6.5: Specific case in which the secret cannot be recovered

$(\mathbf{k}, n)$	Participants
({1,3,4},5)	$P_0, P_1 \in \mathcal{U}_0, P_3 \in \mathcal{U}_1, P_4 \in \mathcal{U}_2$
({1,3},7)	$P_0, P_2 \in \mathcal{U}_0, P_6 \in \mathcal{U}_1$
({2,5},13)	$P_0, P_2, P_7 \in \mathcal{U}_0, P_11, P_12 \in \mathcal{U}_1$
({1,3},31)	$P_0, P_11 \in \mathcal{U}_0, P_30 \in \mathcal{U}_1$
({1,3},127)	$P_0, P_1 \in \mathcal{U}_0, P_7 \in \mathcal{U}_1$

this allocation issue is still open.

6.2.5 Security Analysis

Based on Kurihara et al.'s proof approach [13, 27], we present the effects on our $F_W(i,j)$ in Table 6.4. We introduce Lemmas 7 and 8.

Without loss of generality, we may view matrix $\mathbf{G}'$, corresponding to participants who cooperate to recover the secret, as a matrix listed from to the highest-level participants in the recovery set. In other words, the right block matrices of the first $\mathbf{1}$ in each row of $\mathbf{G}'$ are all $\mathbf{1}$'s and the block matrices below those $\mathbf{1}$'s in that row are also all $\mathbf{1}$'s. We then define matrix $\hat{\mathbf{E}}^{(2)}_{(i)} = \mathbf{E}_{(i)} \oplus \mathbf{1}$ and the $n_p \times n_p$ matrix $\hat{\mathbf{H}}_{(i)}$ by adding one row to $\hat{\mathbf{E}}^{(2)}_{(i)}$ along with all rows of $\hat{\mathbf{E}}^{(2)}_{(i)}$. $\hat{\mathbf{H}}_{(i)}$ can be represented as $\hat{\mathbf{H}}_{(i)} = \bigoplus_{j=0, j\neq i}^{n_p-1} \mathbf{L}_j$.

Lemma 7. *Under Definition 3, Let* $2 \leq L \leq k$. $\mathbf{M} = [\mathbf{P}\ \mathbf{Q}]$, *corresponding to* $\mathbf{D} = [\mathbf{U}''\ \mathbf{V}'']$, *is represented as* $\mathbf{M}^{(L)}_k = [\mathbf{M}_{(i,j)}]^{L-1\ k-1}_{i=1\ j=1}$. *We define*

$$\mathbf{H}^a_{(i,j)} = \begin{cases} \mathbf{H}_{(a\cdot t_i, a\cdot t_j)} & \text{if } 1 \leq a \leq k-2 \\ \mathbf{H}_{((n_p-1)t_i,(n_p-1)t_j)} & \text{if } a = k-1 \end{cases},$$

$$\hat{\mathbf{H}}^a_{(i)} = \begin{cases} \hat{\mathbf{H}}_{(a\cdot t_i)} & \text{if } 1 \leq a \leq k-2 \\ \hat{\mathbf{H}}_{((n_p-1)t_i)} & \text{if } a = k-1 \end{cases}.$$

Furthermore, we define q_i for the i-th block row of $\mathbf{M}_k^{(L)}$ and view q_0 as q_{L-1}. Let $a+q_a \leq b+q_b, a<b$ for all a,b since we may view $\mathbf{M}_k^{(L)}$ as a matrix listed from to the highest-level participants, and let

$$\mathbf{M}_{(i,j)} = \begin{cases} \mathbf{H}_{(0,i)}^{j} & (i+j+q_i < L) \\ \hat{\mathbf{H}}_{(0)}^{j} & (i+j-q_i \geq L) \end{cases}.$$

If $q_i \leq 0$ for all i, then $\operatorname{rank}(\mathbf{M}_k^{(L)}) = \operatorname{rank}(\mathbf{D}) = (L-1)(n_p-1)$, *else* $\operatorname{rank}(\mathbf{M}_k^{(L)}) = \operatorname{rank}(\mathbf{D}) < (L-1)(n_p-1)$.

Proof. For example, if $L = 3 < k, q_1 = 0$, and $q_2 = 0$, $\mathbf{M}_k^{(L)}$ is represented as

$$\mathbf{M}_k^{(L)} = \left.\begin{bmatrix} \mathbf{H}_{(0,k-2)}^{1} & \hat{\mathbf{H}}_{(0)}^{2} & \cdots & \hat{\mathbf{H}}_{(0)}^{k-2} & \hat{\mathbf{H}}_{(0)}^{k-1} \\ \hat{\mathbf{H}}_{(0)}^{1} & \hat{\mathbf{H}}_{(0)}^{2} & \cdots & \hat{\mathbf{H}}_{(0)}^{k-2} & \hat{\mathbf{H}}_{(0)}^{k-1} \end{bmatrix}\right\} L-1 \text{ blocks}$$

and this lemma proves $\operatorname{rank}(\mathbf{M}_k^{(L)}) = (L-1)(n_p-1)$. If $L = k$ and $q_i = 0$ for all i, $\mathbf{M}_k^{(L)}$ is represented as

$$\mathbf{M}_k^{(L)} = \left.\begin{bmatrix} \mathbf{H}_{(0,1)}^{1} & \mathbf{H}_{(0,1)}^{2} & \cdots & \mathbf{H}_{(0,1)}^{k-2} & \hat{\mathbf{H}}_{(0)}^{k-1} \\ \mathbf{H}_{(0,2)}^{1} & \mathbf{H}_{(0,2)}^{2} & \cdots & \hat{\mathbf{H}}_{(0)}^{k-2} & \hat{\mathbf{H}}_{(0)}^{k-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{H}_{(0,k-2)}^{1} & \hat{\mathbf{H}}_{(0)}^{2} & \cdots & \hat{\mathbf{H}}_{(0)}^{k-2} & \hat{\mathbf{H}}_{(0)}^{k-1} \\ \hat{\mathbf{H}}_{(0)}^{1} & \hat{\mathbf{H}}_{(0)}^{2} & \cdots & \hat{\mathbf{H}}_{(0)}^{k-2} & \hat{\mathbf{H}}_{(0)}^{k-1} \end{bmatrix}\right\} L-1 \text{ blocks}$$

and this lemma also proves $\operatorname{rank}(\mathbf{M}_k^{(L)}) = (L-1)(n_p-1)$.

Let $i_0 = 0, \cdots, L-2$ and let $\bar{\mathbf{M}}_k^{(L)}$ be the matrix such that the j-th block column of $\mathbf{M}_k^{(L)}$ for $j = 1, \cdots, k-1$ is switched with the $(k-j)$-th block column and the i-th block row for $i = 1, \cdots, L-2$ is switched with the $(L-1)$-th block row. Since elementary matrix operations do not change the rank of a matrix, we may discuss $\operatorname{rank}(\bar{\mathbf{M}}_k^{(L)})$

instead of $\mathrm{rank}(\mathbf{M}_k^{(L)})$. If $q_{i_0} = 0$ for all i_0, $\bar{\mathbf{M}}_k^{(L)}$ is represented as

$$\bar{\mathbf{M}}_k^{(L)} = \left[\begin{array}{ccccc} \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(0)}^{2} & \hat{\mathbf{H}}_{(0)}^{1} \\ \hline \hat{\mathbf{H}}_{(0)}^{k-1} & \mathbf{H}_{(0,1)}^{k-2} & \cdots & \mathbf{H}_{(0,1)}^{2} & \mathbf{H}_{(0,1)}^{1} \\ \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \cdots & \mathbf{H}_{(0,2)}^{2} & \mathbf{H}_{(0,2)}^{1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(0)}^{2} & \mathbf{H}_{(0,k-2)}^{1} \end{array}\right] \Bigg\} L-1 \text{ blocks} .$$

Since $\hat{\mathbf{H}}_{(i)}^{x} \oplus \mathbf{H}_{(i,j)}^{x} = \hat{\mathbf{H}}_{(j)}^{x}$, $\bar{\mathbf{M}}_k^{(L)}$ can be transformed to

$$\bar{\mathbf{M}}_k^{(L)} \to \left[\begin{array}{ccccc} \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \hat{\mathbf{H}}_{(0)}^{k-3} & \cdots & \hat{\mathbf{H}}_{(0)}^{1} \\ \hline \boldsymbol{O} & \hat{\mathbf{H}}_{(1)}^{k-2} & \hat{\mathbf{H}}_{(1)}^{k-3} & \cdots & \hat{\mathbf{H}}_{(1)}^{1} \\ \boldsymbol{O} & \boldsymbol{O} & \hat{\mathbf{H}}_{(2)}^{k-3} & \cdots & \hat{\mathbf{H}}_{(2)}^{1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \cdots & \hat{\mathbf{H}}_{(k-2)}^{1} \end{array}\right] = \left[\begin{array}{ccc} \mathbf{M}_0 & * & * \\ \hline \boldsymbol{O} & & \\ \vdots & * & \mathbf{M}' \\ \boldsymbol{O} & & \end{array}\right]$$

by using elementary row operations. Here, we define matrices $\mathbf{M}_0$ and $\mathbf{M}'$. Since $\hat{\mathbf{H}}_{(i_0)}^{k-1-i_0} \neq \boldsymbol{O}$ for all i_0, $\mathrm{rank}(\bar{\mathbf{M}}_k^{(L)}) = (L-1)(n_p-1)$. If $q_{i_0} \geq 1$ for at least any one of i_0, $\mathrm{rank}(\bar{\mathbf{M}}_k^{(L)}) < (L-1)(n_p-1)$ because at least one of the diagonal block matrices of $\bar{\mathbf{M}}_k^{(L)}$, corresponding to $\hat{\mathbf{H}}_{(i_0)}^{k-1-i_0}$ for all i_0, is $\boldsymbol{O}$. Note that if $L < k$, we can discuss the diagonal block matrices of $\mathbf{M}'$.

Next, we consider the remaining condition $q_{i_0} < 0$ for all i_0. For example, if $q_1 = -1, q_2 = -2$, and the others $q_{i_0} = 0$, $\bar{\mathbf{M}}_k^{(L)}$ can be transformed to

$$\bar{\mathbf{M}}_k^{(L)} \to \left[\begin{array}{cccccc} \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \hat{\mathbf{H}}_{(0)}^{k-3} & \hat{\mathbf{H}}_{(0)}^{k-4} & \cdots & \hat{\mathbf{H}}_{(0)}^{1} \\ \hline \hat{\mathbf{H}}_{(1)}^{k-1} & \hat{\mathbf{H}}_{(1)}^{k-2} & \hat{\mathbf{H}}_{(1)}^{k-3} & \hat{\mathbf{H}}_{(1)}^{k-4} & \cdots & \hat{\mathbf{H}}_{(1)}^{1} \\ \hat{\mathbf{H}}_{(2)}^{k-1} & \hat{\mathbf{H}}_{(2)}^{k-2} & \hat{\mathbf{H}}_{(2)}^{k-3} & \hat{\mathbf{H}}_{(2)}^{k-4} & \cdots & \hat{\mathbf{H}}_{(2)}^{1} \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \hat{\mathbf{H}}_{(3)}^{k-4} & \cdots & \hat{\mathbf{H}}_{(3)}^{1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \cdots & \hat{\mathbf{H}}_{(k-2)}^{1} \end{array}\right] .$$

In general, $\bar{\mathbf{M}}_k^{(L)}$ can be transformed to

$$\bar{\mathbf{M}}_k^{(L)} \to \left[\begin{array}{cccc} \mathbf{M}_0^{(m_0)} & & & * \\ & \mathbf{M}_1^{(m_1)} & & \\ & & \ddots & \\ \boldsymbol{O} & & & \mathbf{M}_\alpha^{(m_\alpha)} \end{array}\right],$$

where $\mathbf{M}_i^{(m_i)}$ is an $m_i \times m_i$ block matrix. If $\mathrm{rank}(\mathbf{M}_i^{(m_i)}) = m_i(n_p - 1)$ for all i, $\mathrm{rank}(\bar{\mathbf{M}}_k^{(L)}) = (L-1)(n_p-1)$. Without loss of generality, we can discuss $\mathbf{M}_0^{(m_0)}$ instead of all matrices $\mathbf{M}_i^{(m_i)}$. $\mathbf{M}_0^{(m_0)}$ can be transformed to

$$\mathbf{M}_0^{(m_0)} = \begin{bmatrix} \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(0)}^{(k-m_0)} \\ \hat{\mathbf{H}}_{(1)}^{k-1} & \hat{\mathbf{H}}_{(1)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(1)}^{(k-m_0)} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{H}}_{(m_0-1)}^{k-1} & \hat{\mathbf{H}}_{(m_0-1)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(m_0-1)}^{(k-m_0)} \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(0)}^{(k-m_0)} \\ \hline \mathbf{H}_{(0,1)}^{k-1} & \mathbf{H}_{(0,1)}^{k-2} & \cdots & \mathbf{H}_{(0,1)}^{(k-m_0)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{(0,m_0-1)}^{k-1} & \mathbf{H}_{(0,m_0-1)}^{k-2} & \cdots & \mathbf{H}_{(0,m_0-1)}^{(k-m_0)} \end{bmatrix} = \begin{bmatrix} * \\ \hline \\ \mathbf{H}' \\ \\ \end{bmatrix}.$$

Here, let $\mathbf{M}$ be the matrix used in Kurihara et al.'s scheme and let $\bar{\mathbf{M}}$ be the matrix such that the j-th block column of $\mathbf{M}$ for $j = 1, \cdots, k-1$ is switched with the $(k-j)$-th block column. $\bar{\mathbf{M}}$ can be also transformed to the same matrix transformed from $\mathbf{M}$. Then, we can view $\mathbf{H}'$ as a submatrix of $\bar{\mathbf{M}}$. $\mathbf{H}'$ can be transformed to the row echelon form as with $\bar{\mathbf{M}}$ or $\mathbf{M}$. In other words, $\mathrm{rank}(\mathbf{H}') = (m_0 - 1)(n_p - 1)$. Furthermore, $\mathrm{rank}(\mathbf{M}_0^{(m_0)}) = m_0(n_p - 1)$ because each block row of $\mathbf{H}'$ is XOR-ed with the first block row of $\mathbf{M}_0^{(m_0)}$. Therefore, $\mathrm{rank}(\bar{\mathbf{M}}_k^{(L)}) = (L-1)(n_p-1)$ and the proof is complete. □

Lemma 8. *Under Definition 3, Let* $2 \leq L$. *If the block matrices in* $\mathbf{V}$ *are all* $\mathbf{1}$*'s,* $\mathrm{rank}(\mathbf{G}') \leq (L-1)(n_p-1)$. *Furthermore, if the block matrices in* $\mathbf{V}$ *and the* $k-2, k-1, \cdots, j$*-th block columns of* $\mathbf{U}$ *are all* $\mathbf{1}$*'s,* $\mathrm{rank}(\mathbf{G}') \leq (L-1)(n_p-1)$.

Proof. Matrices $\mathbf{V}$ and $\mathbf{V}'$ are represented as

$$\mathbf{V} = \left.\begin{bmatrix} \mathbf{1} \\ \vdots \\ \mathbf{1} \end{bmatrix}\right\} L \text{ blocks}, \quad \mathbf{V}' = \begin{bmatrix} \mathbf{1} \\ \hline \boldsymbol{O} \\ \vdots \\ \boldsymbol{O} \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \hline \mathbf{V}'' \end{bmatrix}.$$

Here, we refer to Lemma 7. $\mathbf{M}$, corresponding to $\mathbf{D} = [\mathbf{U}''\ \mathbf{V}'']$, can be represented as $\mathbf{M}_k^{(L)}$ in which $\mathbf{M}_0$ is always $\boldsymbol{O}$ if $L \leq k$. It is apparent that $\mathrm{rank}(\mathbf{M}) \leq (L-1)(n_p-1)$.

Furthermore, even if $L > k$, we can retain $\text{rank}(\mathbf{M}) \leq (L-1)(n_p-1)$. The proof is therefore complete. □

We introduce Theorems 4 and 5 under the access structure of Definition 1. Theorems 4 and 5 show perfect privacy $H(S|S_T) = H(S)$.

Theorem 4. *Let $1 \leq L \leq k-1$ in Definition 3. Assume that any set of L participants $T = \{P_{t_0}, \cdots, P_{t_{L-1}}\}$ agrees to recover the secret. Then, we receive no information regarding the secret.*

Proof. Consider $\mathbf{G}'$ corresponding to the set of participants. Let $\alpha = L'(n_p-1)$ be the rank of $\mathbf{G}'$. If $L = 1$, $\text{rank}(\mathbf{G}') = L(n_p-1) = \alpha$. From Lemma 7, if $L = 2, \cdots, k-1$, $\text{rank}(\mathbf{G}') = \alpha \leq L(n_p-1)$. In other words, we can find $\bar{\mathbf{G}}' = \left[\bar{\mathbf{U}}\ \bar{\mathbf{V}}\right]$ such that $\text{rank}(\bar{\mathbf{G}}') = \alpha$ and $\text{rank}(\bar{\mathbf{U}}) = \alpha$. If we receive no information regarding the secret from the $\bar{\mathbf{G}}'$, we do not receive information more than $\bar{\mathbf{G}}'$.

Suppose that elements of $\mathbf{s}$, as given in Table 6.1, and elements of $\mathbf{r}$ are mutually independent and that elements of $\mathbf{r}$ are selected from the finite set $\{0,1\}^d$ with uniform probability $1/2^d$. $\bar{\mathbf{U}}$ and $\bar{\mathbf{V}}$ are $L'(n_p-1) \times ((k-1)n_p-1)$ and $L'(n_p-1) \times n_p$ matrices, respectively. Because all rows of $\bar{\mathbf{U}}$ are linearly independent for $1 \leq L' \leq k-1$, we can use Lemma 6. In other words, all elements of the $L'(n_p-1)$-dimensional vector $\bar{\mathbf{U}} \cdot \mathbf{r}$ are d-bit random numbers that are mutually independent and distributed uniformly over $\{0,1\}^d$. Therefore, the vector $\bar{\mathbf{U}} \cdot \mathbf{r}$ is uniformly distributed over $\{0,1\}^{dL'(n_p-1)}$.

Next, we suppose that $\mathbf{w}'$ denotes a fixed value of $\mathbf{w}$. $\mathbf{w} = \mathbf{w}'$ can be obtained with uniform probability $(1/2)^{dL'(n_p-1)}$ from any selected $\mathbf{s}$ or $\bar{\mathbf{V}} \cdot \mathbf{s}'$. Because $\mathbf{s}$ is independent of $\mathbf{w}$, we have $H(S|S_T) = H(S)$ and therefore receive no information regarding the secret. □

In hierarchical schemes, we need to consider the case of k or more participants in an unauthorized set T.

Theorem 5. *Let $L \geq k$ in Definition 3. Assume that any set of L participants $T = \{P_{t_0}, \cdots, P_{t_{L-1}}\}$ agrees to recover the secret. Then, the secret cannot be recovered.*

Proof. From Lemma 8, when no participants at the highest level $\mathcal{U}_0$ cooperate to recover the secret, the secret cannot be recovered. Next, from Lemma 7, when a minimal number of higher-level participants are not included, $\text{rank}(\mathbf{G}') < k(n_p - 1)$. If $L > k$, each of the $L-k$ block rows should include $\hat{\mathbf{H}}^j_{(0)}$, shown in Lemma 7, because the block row is not for the participant at the highest level $\mathcal{U}_0$. As a result, we can retain $\text{rank}(\mathbf{G}') < k(n_p - 1)$ even if $L > k$. Therefore, we can simplify the proof to the case of $L < k$ shown in Theorem 4 and receive no information regarding the secret. □

Consider correctness $H(S|S_B) = 0$. Let $L = k$ in Definition 3. Assume that any set of k participants $B = \{P_{t_0}, \cdots, P_{t_{k-1}}\}$ agrees to recover the secret. Then, Lemma 4 holds even if matrix $\mathbf{1}$'s are included and the secret can be recovered under some condition. Here, consider the set of participants that can recover the secret. Because the corresponding $\mathbf{G}'$ has $\text{rank}(\mathbf{G}') = k(n_p - 1)$ and $\text{rank}(\mathbf{U}) = (k-1)(n_p - 1)$ from Lemma 7, the secret is therefore recovered. However, the condition is still open because depending on the allocation of participant identities, there is a specific case in which the secret cannot be recovered in spite of the presence of an authorized subset, as shown in Section 6.2.4.

Under some condition for correctness, we conclude that both correctness and perfect privacy hold. Because every bit size of shares equals the bit size of the secret, our scheme is *ideal*.

6.2.6 Brief Example of Our Scheme

We consider a $(\{2,3\},5)$ HSSS as an example. Let participants $P_0, P_1, P_2 \in \mathcal{U}_0$, and $P_3, P_4 \in \mathcal{U}_1$. We generate each share $w_i = w_{(i,0)}||\cdots||w_{(i,3)}$ for participant P_i with $\mathbf{w} = \mathbf{G} \cdot \begin{bmatrix} \mathbf{r} \\ \mathbf{s} \end{bmatrix}$. For example, we can view the $\mathbf{v}_{(0,0)}$ used to generate $w_{(0,0)}$ as $\mathbf{v}_{(0,0)} =$

(1000 10000 0000), i.e.,

$$\mathbf{w} = \begin{bmatrix} \underline{\mathbf{w}_0} \\ \underline{\mathbf{w}_1} \\ \underline{\mathbf{w}_2} \\ \underline{\mathbf{w}_3} \\ \underline{\mathbf{w}_4} \end{bmatrix} = \begin{bmatrix} w_{(0,0)} \\ w_{(0,1)} \\ w_{(0,2)} \\ \underline{w_{(0,3)}} \\ w_{(1,0)} \\ w_{(1,1)} \\ \vdots \\ w_{(3,2)} \\ \underline{w_{(3,3)}} \\ w_{(4,0)} \\ w_{(4,1)} \\ w_{(4,2)} \\ w_{(4,3)} \end{bmatrix} = \mathbf{G} \cdot \begin{bmatrix} \mathbf{r} \\ \mathbf{s} \end{bmatrix} = \begin{bmatrix} 1000\ 10000\ 0000 \\ 0100\ 01000\ 1000 \\ 0010\ 00100\ 0100 \\ \underline{0001\ 00010\ 0010} \\ 1000\ 01000\ 0001 \\ 0100\ 00100\ 0000 \\ \vdots \\ 0010\ 10000\ 1111 \\ \underline{0001\ 01000\ 1111} \\ 1000\ 00001\ 1111 \\ 0100\ 10000\ 1111 \\ 0010\ 01000\ 1111 \\ 0001\ 00100\ 1111 \end{bmatrix} \begin{bmatrix} r_0^0 \\ r_1^0 \\ r_2^0 \\ r_3^0 \\ r_0^1 \\ r_1^1 \\ r_2^1 \\ r_3^1 \\ r_4^1 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}.$$

When participants P_1, P_2, P_4 cooperate to recover the secret, the secret can be recovered because

$$[\mathbf{G}\ \mathbf{I}_{12}] = \begin{bmatrix} 1000\ 01000\ 0001 & 1000\ 0000\ 0000 \\ 0100\ 00100\ 0000 & 0100\ 0000\ 0000 \\ 0010\ 00010\ 1000 & 0010\ 0000\ 0000 \\ \underline{0001\ 00001\ 0100} & \underline{0001\ 0000\ 0000} \\ 1000\ 00100\ 0010 & 0000\ 1000\ 0000 \\ 0100\ 00010\ 0001 & 0000\ 0100\ 0000 \\ 0010\ 00001\ 0000 & 0000\ 0010\ 0000 \\ \underline{0001\ 10000\ 1000} & \underline{0000\ 0001\ 0000} \\ 1000\ 11111\ 1111 & 0000\ 0000\ 1000 \\ 0100\ 11111\ 1111 & 0000\ 0000\ 0100 \\ 0010\ 11111\ 1111 & 0000\ 0000\ 0010 \\ 0001\ 11111\ 1111 & 0000\ 0000\ 0001 \end{bmatrix},$$

$$[\mathbf{G}\ \mathbf{I}_{12}] \to \left[\bar{\mathbf{G}}\ \mathbf{J}\right] \to \left[\bar{\bar{\mathbf{G}}}\ \bar{\bar{\mathbf{J}}}\right] = \begin{bmatrix} 1000\ 01000\ 0001 & 1000\ 0000\ 0000 \\ 0100\ 00100\ 0000 & 0100\ 0000\ 0000 \\ 0010\ 00010\ 1000 & 0010\ 0000\ 0000 \\ \underline{0001\ 00001\ 0100} & \underline{0001\ 0000\ 0000} \\ 0000\ 11101\ 1111 & 1001\ 1001\ 0000 \\ 0000\ 01010\ 0010 & 1100\ 1100\ 0000 \\ 0000\ 00101\ 1001 & 0110\ 0110\ 0000 \\ \underline{0000\ 00011\ 1000} & \underline{0010\ 0010\ 0000} \\ 0000\ 00000\ \underline{1000} & \underline{1011\ 1001\ 0010} \\ 0000\ 00000\ \underline{0100} & \underline{0001\ 0010\ 0011} \\ 0000\ 00000\ \underline{0010} & \underline{0100\ 1000\ 1100} \\ 0000\ 00000\ \underline{0001} & \underline{1001\ 1101\ 0100} \end{bmatrix},$$

$$\mathbf{G}' = \begin{bmatrix} \mathbf{I}_4 & \mathbf{E}_{(1)} & \mathbf{E}_{(4)} \\ \mathbf{I}_4 & \mathbf{E}_{(2)} & \mathbf{E}_{(3)} \\ \mathbf{I}_4 & \mathbf{1} & \mathbf{1} \end{bmatrix}, \ \mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 1011\ 1001\ 0010 \\ 0001\ 0010\ 0011 \\ 0100\ 1000\ 1100 \\ 1001\ 1101\ 0100 \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \mathbf{w}_4 \end{bmatrix}.$$

However, participants P_0, P_3, P_4 cannot recover the secret.

6.3 Software Implementation

For the $(\mathbf{k}, n)$ HSSS, we evaluated our scheme in the test environment, as described in Section 2.11. Table 6.6 presents our experimental results. We used $d = 64$ and used Xorshift for random number generation. The processing time to generate random numbers was included in the results of distribution. The implementation used in our experiments can be applied to any level, not specialized for, for example, $\mathbf{k} = \{1, 3\}$. We then assigned n participants such that participants in each level could cooperate to recover the secret and most of the remaining $n-k$ participants belonged to lower levels, especially the lowest level. We obtained almost the same results in both $(\{1, k\}, n)$ and $(\{2, 3, k\}, n)$ with $k = 5$ and $n = 11$.

Table 6.6: Experimental results

$(\mathbf{k}, n)$	Distribution (Mbps)	Recovery (Mbps)
({1,3},5)	1,390	6,750
({1,3},11)	130	1,720
({1,3},59)	4.07	206
({1,5},11)	62.6	1,070
({2,4},7)	321	2,340
({2,3,5},11)	62.5	1,090
({2,4,6,10},17)	11.5	195
({3,7,11,14,17},23)	3.52	34.7

6.4 Conclusions of This Chapter

We focused on a fast $(\mathbf{k}, n)$ HSSS that also supports efficient deletion of the secret. To achieve this, we applied a hierarchy to Kurihara et al.'s XOR-based SSS and found that all elements corresponding to the secret and some random values of the binary generator matrix are set to 1 for the lower-level participants. In addition, we showed that the traditional allocation issue with participant identities could be removed. Given that the implementation used in our experiments can be applied to any level, we found that our implementation was able to recover a given secret with $\mathbf{k} = \{1, 3\}$ and $n = 5$ at a processing speed of approximately 6.75 Gbps.

Considering the scheme constructed by our HSSS with one level, it has the same performance as Kurihara et al.'s *ideal* (k, n) threshold SSS. In other words, when k participants at the highest level $\mathcal{U}_0$ cooperate to recover the secret, our scheme can yield the same result as Kurihara et al.'s *ideal* (k, n) threshold SSS. Therefore, our HSSS does not have any degradation by achieving hierarchy.

CHAPTER 7

Computational Costs

7.1 Tassa's Approach

We focus on the comparison between Tassa's Approach and our HSSS based on information dispersal techniques, shown in Chapter 4. Table 7.1 shows the computational costs of the recovery algorithm. In general, the size of the secret, for example, a file size

Table 7.1: 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

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 7.2 shows example results in terms of the number of recovery operations in a $(\{1,3\},n)$ HSSS. In the table, the numbers in parentheses show operations not including ones related to either determinant or matrix inverse. Furthermore, an asterisk (*) identifies operations related to identities. Note that the size of 888,710 bytes was padded to 888,711 bytes for our scheme. From the table, we observe that our scheme had fewer division operations and was faster without optimization. We viewed these costs as operations over GF(2^L) although Tassa's scheme can be applied only to GF(p).

Table 7.2: The number of recovery operations for an input of size 888,710 bytes

	Tassa's approach	Our scheme
Addition	7,983,659 (0)	7,109,724 (7,109,688)
Multiplication	10,622,853 (12*)	7,998,429 (7,998,399)
Division	3,544,146 (888,710)	13 (0)
Determinant	888,711	0
Matrix inverse	0	1

7.2 Blömer et al.'s Technique

Blömer et al. [17] define the matrix representation of finite fields as Definition 5. Given $f=(1,1,1,1,0,0,0,0)^{\mathrm{T}}, g=(1,1,0,0,1,1,1,1)^{\mathrm{T}} \in \mathrm{GF}(2^8)$ as an example, we obtain

$$\tau(f)=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad \tau(f)\cdot g=\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

where the irreducible polynomial is $X^8 + X^4 + X^3 + X^2 + 1$. Since we may view the coefficient vector as a hexadecimal number of $f_{L-1} \cdots f_1 f_0$ binary, $0\mathrm{Fh} \cdot \mathrm{F3h} = 28\mathrm{h}$.

Next, we consider applying Blömer et al.'s technique to our scheme because in [15], Kurihara et al. reported that any SSS over $\mathrm{GF}(2^L)$ could be converted to an SSS over $\mathrm{GF}(2)$. As they described in Section 4 of [17] in terms of implementation, we stored all coefficient vectors of field elements in a table and used table look-ups to compute τ. Therefore, 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 $\mathrm{GF}(2^8)$, a multiplication operation only required $\mathcal{O}(1)$.

Definition 5. *Elements in* $\mathrm{GF}(2^L)$ *can be identified with polynomials*

$$f_L(X) = \sum_{i=0}^{L-1} f_i X^i, f_i \in \mathrm{GF}(2)$$

and let the coefficient vector of the element $f_L(X)$ *be column vector* $(f_0, \cdots, f_{L-1})^{\mathrm{T}}$. *Then, for any* $f \in \mathrm{GF}(2^L)$, *let* $\tau(f)$ *be the matrix whose* i*-th column is the coefficient vector of* $X^{i-1} f \pmod{p(X)}$, *where* $p(X)$ *is an irreducible polynomial of degree* L *in* $\mathrm{GF}(2)[X]$.

7.3 Our $(\{1, k\}, k + 1)$ Optimized HSSS Based on Information Dispersal Techniques

In our scheme, we can construct a scheme using only XOR operations in terms of optimization because we can use $(k + 1) \times k$ 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 generator matrix requires no multiplication operations. We then used simple 64-bit XOR operations instead of $\mathrm{GF}(2^L)$. With $k = 3$, we achieved approximately 6.3 Gbps

in the result of recovery. We observe that our $(\{1, k\}, k+1)$ HSSS was much faster than Tassa's approach.

7.4 Our XOR-based HSSS

Table 7.3 shows the computational costs. In general, the size of the secret exceeded $d(n_p - 1)$ bits. In our analysis, we refer to such an initial computation processed once

Table 7.3: Computational costs

	Precomputation	Distribution	Recovery
Our scheme	$\mathcal{O}(k^3 n_p^3)$	$\mathcal{O}(kn)\|s\| \leq x \leq \mathcal{O}(kn_p n)\|s\|$	$\mathcal{O}(kn_p)\|s\|$
Tassa's scheme	$\mathcal{O}(k^3)$	$\mathcal{O}(kn)$	$\mathcal{O}(k^3)$

- k is the maximal threshold, shown in Definition 1.
- $|s|$ denotes the bit length of the secret, i.e., $|s| = d(n_p - 1)$.
- A $t \times t$ determinant required $\mathcal{O}(t^3)$ when we used LU decomposition.
- Our scheme shows the costs of the bitwise XOR operations, and Tassa's scheme shows the costs of arithmetic operations.

for that recovery as a precomputation. We may view the cost of recovery in our scheme as the one in Kurihara et al.'s scheme [13, 14] because no changes were made to them in terms of the number of necessary XOR operations. As for the cost of distribution, when all n participants belong to the highest level, our algorithm in Table 6.4 has the minimum computational cost. More specifically, it requires $\mathcal{O}(kn)|s|$ bitwise XOR operations to generate n shares because our scheme can yield the same result as Kurihara et al.'s (k, n) threshold scheme, which required $(k - 1 - \frac{1}{n_p})|s|$ as the average number of XOR operations to generate one share. Next, when more participants belong to the lowest level, our algorithm has the maximum computational cost. More Specifically, it requires $(n_p(k-1)-1)n|s| = \mathcal{O}(kn_p n)$ bitwise XOR operations to generate n shares. We assume that all n participants belong to the lowest level although no sets can recover the secret.

Our scheme requires

$$(k-1+\{(k-1)(n_p-1)-1\})|s| = (n_p(k-1)-1)|s|$$

as the number of XOR operations to generate one share at the lowest level. In Table 7.3, we can see that our scheme is more efficient than Tassa's approach if n_p is not extremely large. Note that the precomputation cost is not dominant in a large file, while our scheme is not more efficient than Tassa's approach. Furthermore, as shown in Section 3.6, we can see that arithmetic operations required high computational costs.

7.5 Best HSSS for Performance

In this research, we presented four approaches, including three HSSSs applicable at any level, but we discuss what scheme is the best for performance because the XOR-based HSSS depends on the size of n_p for recovery, as shown in Table 7.3. In the HSSS over finite fields of characteristic two, we achieved approximately 97 Mbps with any n, as shown in Table 3.6. Considering the implementation optimized for a $(\{1,3\},n)$ HSSS for recovery, we then achieved approximately 970 Mbps with any n. In the HSSS based on information dispersal techniques, we achieved approximately 850 Mbps with any n, as shown in Table 4.5. The implementation is almost optimized. Table 7.4 shows additional experimental results, taken as the average of 800 experiments, for the

Table 7.4: Additional experimental results

$(\mathbf{k},n)$	Recovery (Mbps)
({1,3}, 5)	8,065
({1,3},11)	1,749
({1,3},13)	1,645
({1,3},17)	822
({1,3},19)	713
({1,3},29)	421
({1,3},43)	254
({1,3},59)	175
({1,3},109)	99.0

XOR-based HSSS. The implementation is also almost optimized.

Operations over GF(2^L) and only simple XOR operations naturally motivated us to consider fast schemes because these operations yield faster schemes, as shown in Section 1.4. Chen et al.'s scheme [18] caused us to consider using IDAs because they proposed an SSS that can be implemented using only XOR operations. Needless to say, we require fast schemes that can be applied to any level as a basic policy to achieve hierarchy. Table 7.5 shows the experimental results of the four HSSSs for recovery with $\mathbf{k} = \{1, 3\}$ and any n. Here, we focus on the three HSSSs applicable at any level. In

Table 7.5: Recovery with $\mathbf{k} = \{1, 3\}$ (Mbps)

n	Table 3.6	Table 4.5	Table 5.13	Table 7.4
5	970 (97.07)	857	7060 $\sim$ 8370	8,065
11				1,749
13				1,645
17				822
19				713
29				421
43				254
59				175
109				99.0

- Table 3.6: HSSS over finite fields of characteristic two
- Table 4.5: HSSS based on information dispersal techniques
- Table 5.13: XOR-based HSSS for a small number of indispensable participants
- Table 7.4: XOR-based HSSS

terms of computational efficiency, we can see that the XOR-based HSSS is more efficient than the optimized HSSS over finite fields of characteristic two if $n \leq 13$ and the HSSS over finite fields of characteristic two if $n \leq 109$.

CHAPTER 8

Conclusions

We focused on fast HSSSs in which a minimal number of higher-level participants are required for recovery of the secret. They are conjunctive HSSSs and also supports efficient deletion of the secret. To achieve this, we presented four approaches. First, we presented a HSSS applicable at any level over finite fields of characteristic two. Next, we applied the general concept of hierarchy to the generator matrix used in a systematic IDA and proposed a HSSS applicable at any level. Then, we proposed a $(\{1,3\},n)$ XOR-based HSSS for a small number of indispensable participants specially made for three participants including one or two indispensable participants to recover the secret. Finally, we applied the general concept of hierarchy to Kurihara et al.'s XOR-based SSS and proposed a HSSS applicable at any level.

Given that the implementation used in our experiments can be applied to any level, we found that our implementation was able to recover a given secret with $\mathbf{k} = \{1,3\}$ at a processing speed of approximately 97 Mbps with any n for the first approach, 850 Mbps with any n for the second approach, and 8.0 Gbps with $n = 5$ for the fourth approach. Then, under the implementation optimized for a $(\{1,3\},n)$ HSSS for recovery, we achieved approximately 970 Mbps with any n for the first approach, 6.3 Gbps with $n = 4$ for the second approach, and 8.3 Gbps with $n = 5$ for the third approach.

In hierarchical schemes, including Tassa's scheme, depending on the allocation of participant identities, there is a specific case in which the secret cannot be recovered in spite of the presence of an authorized subset. The allocation issue could be removed by using our XOR-based HSSS that can be applied to any level, but is still open.

Bibliography

[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] Kothari, S.: Generalized Linear Threshold Scheme. Advances in Cryptology - CRYPTO '84, LNCS 196, 231–241 (1985)

[4] Ito, M., Saito, A., and Nishizeki, T.: Secret Sharing Scheme Realizing General Access Structure. IEEE Global Telecommun. Conf., Globecom '87, 99–102 (1987)

[5] Ito, M., Saito, A., and Nishizeki, T.: Secret Sharing Scheme Realizing General Access Structure. IEICE Trans. Fundamentals, Vol.J71-A, No.8, 1592–1598 (1988) (in Japanese) ⟨https://ci.nii.ac.jp/naid/40004637233⟩

[6] Ito, M., Saito, A., and Nishizeki, T.: Secret Sharing Scheme Realizing General Access Structure. (in Japanese) ⟨https://dbnst.nii.ac.jp/pro/detail/319⟩

[7] Ghodosi, H., Pieprzyk, J., and Safavi-Naini, R.: Secret Sharing in Multilevel and Compartmented Groups. ACISP 1998, LNCS 1438, 367–378 (1998)

[8] Tassa, T.: Hierarchical Threshold Secret Sharing. TCC 2004, LNCS 2951, 473–490 (2004)

[9] Tassa, T.: Hierarchical Threshold Secret Sharing. Journal of Cryptology, Vol.20, No.2, 237–264 (2007)

[10] Kurihara, J., Tanaka, T., Kiyomoto, S.: (No English title). Japan patent JP2009-288338A, Japan patent JP5241325B, 2008-5-27 (in Japanese)

[11] Fujii, Y., Tada, M., Hosaka, N., Tochikubo, K., Kato, T.: A Fast $(2, n)$-Threshold Scheme and Its Application. CSS 2005, 631–636 (2005) (in Japanese)

[12] 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)

[13] 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)

[14] 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)

[15] 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)

[16] 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)

[17] 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)

[18] Chen, L., Laing, T.M., Martin K.M.: Efficient, XOR-Based, Ideal (t, n) threshold Schemes. CANS 2016, LNCS 10052, 467–483 (2016)

[19] Selçuk, A.A., Kaşkaloğlu, K., Özbudak, F.: On Hierarchical Threshold Secret Sharing. IACR Cryptology ePrint Archive 2009, 450 (2009)

[20] Simmons, G.J.: How to (really) share a secret. Advances in Cryptology - CRYPTO '88, LNCS 403, 390–448 (1990)

[21] Brickell, E.F.: Some ideal secret sharing schemes. Advances in Cryptology - EUROCRYPT '89, LNCS 434, 468–475 (1990)

[22] Castiglione, A., De Santis, A. and Masucci, B.: Hierarchical and shared key assignment. NBiS-2014, 263–270 (2014)

[23] 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, 850–865 (2016)

[24] Beimel, A.: Secret-Sharing Schemes, A Survey. IWCC 2011, LNCS 6639, 11–46 (2011)

[25] Blundo, C., De Santis, A., Gargano, L., Vaccaro, U.: On the information rate of secret sharing schemes. TCS, Vol.154, 283–306 (1996)

[26] 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)

[27] 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)

[28] Farràs, O., Padró, C.: Ideal Hierarchical Secret Sharing Schemes. TCC 2010, LNCS 5978, 219–236 (2010)

[29] Matsuo, M. and Mutou, K.: (k, n)-Threshold Secret Sharing Scheme Using Exclusive OR. Panasonic Technical Journal, Vol.59, No.2 (2013) (in Japanese)

[30] Suga, Y.: New constructions of (k, n)-threshold secret sharing schemes using exclusive-OR operations and their advantages. CSS 2012, 185–192 (2012) (in Japanese)

[31] Suga, Y.: New Constructions of $(2, n)$-Threshold Secret Sharing Schemes Using Exclusive-OR Operations. 2013 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing rm (IMIS), 837–842 (2013)

[32] Suga, Y.: An algebraic interpretation of the XOR-based secret sharing schemes. SCIS 2018, 2A3-5 (2018) (in Japanese)

[33] Ke, C., Anada, H., Kawamoto, J., Morozov, K. and Sakurai, K.: Cross-group Secret Sharing for Distributed Storages over Providers. SCIS 2016, 3A1-3 (2016) (in Japanese)

[34] Ozaki, H. and Sakurai, K.: Yet another security issue around Secret Sharing Schemes – Revisiting "unfair" cryptosystems. SCIS 2016, 3A1-5 (2016) (in Japanese)

[35] Cianciullo, L., Ghodosi, H.: Improvements to Almost Optimum Secret Sharing with Cheating Detection. IWSEC 2018, LNCS 11049, 193–205 (2018)

[36] Lorentz, G. G., Jetter, K. and Riemenschneider, S. D.: Birkhoff Interpolation. Encyclopedia of Mathematics and its Applications 19 (1983, 1984)

[37] Lorentz, G. G. and Zeller, K. L.: Birkhoff Interpolation. SIAM Journal on Numerical Analysis, Vol.8, No.1, 43–48 (1971)

[38] Käsper, E., Nikov, V. and Nikova, S.: Strongly multiplicative hierarchical threshold secret sharing. Information Theoretic Security, LNCS 4883, 148–168 (2009)

[39] Roy, P.S., Dutta, S., Morozov, K., Adhikari, A., Fukushima, K., Kiyomoto, S., Sakurai, K.: Hierarchical Secret Sharing Schemes Secure Against Rushing Adversary: Cheater Identification and Robustness. ISPEC 2018, LNCS 11125, 578–594 (2018)

[40] 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)

[41] Blakley, G.R., Meadows C.: Security of Ramp Schemes. Advances in Cryptology - CRYPTO '84, LNCS 196, 242–268 (1985)

[42] Krawczyk, H.: Secret Sharing Made Short. Advances in Cryptology - CRYPTO '93, LNCS 773, 136–146 (1993)

[43] 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)

[44] Rivest, R.L.: All-or-nothing encryption and the package transform. FSE 1997, LNCS 1267, 210–218 (1997)

[45] Béguin P., Cresti A.: General Short Computational Secret Sharing Schemes. Advances in Cryptology - EUROCRYPT '95, LNCS 921, 194–208 (1995)

[46] Ikarashi, D., Tsuyuzaki, K., Kawahara, Y.: SHSS:"Super High-speed (or, Sugoku Hayai) Secret Sharing" Library for Object Storage Systems. IPSJ CSEC, 2015-CSEC-70(26), 1–8 (2015) (in Japanese)

[47] Matsumoto, T., Seito, T., Kamoshida, A., Shingai, T. and Sato, A.: High-Speed Secret Sharing System for Secure Data Storage Service. SCIS 2012, 1E2-4, 1–6 (2012) (in Japanese)

[48] Jackson, W.-A., Martin, K.M.: A Combinatorial Interpretation of Ramp Schemes. Australasian Journal of Combinatorics, Vol.14, 51–60 (1996)

[49] 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)

[50] 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)

[51] Shima, K. and Doi, H.: A study on fast hierarchical secret sharing schemes. CSS 2015, 3C4-5, 1327–1334 (2015) (in Japanese)

[52] Shima, K. and Doi, H.: A study on $(\{1, k\}, n)$ hierarchical secret sharing schemes over finite fields of characteristic 2. IPSJ SIG Technical Reports, CSEC 72, No.4 (2016) (in Japanese)

[53] Shima, K. and Doi, H.: A study on hierarchical secret sharing schemes applicable to any level over finite fields of characteristic 2. CSS 2016, 2C2-2, 425–432 (2016) (in Japanese)

[54] Shima, K. and Doi, H.: $(\{1, 3\}, n)$ hierarchical secret sharing scheme based on XOR operations for a small number of indispensable participants. AsiaJCIS 2016, 108–114 (2016) **[Refereed]**

[55] Shima, K. and Doi, H.: A Hierarchical Secret Sharing Scheme over Finite Fields of Characteristic 2. Journal of Information Processing, Vol.25, 875–883 (2017) **[Refereed]**

[56] Shima, K. and Doi, H.: A new construction of hierarchical secret sharing schemes and its evaluation. CSS 2017, 2E1-3, 457–464 (2017) (in Japanese)

[57] Shima, K. and Doi, H.: An XOR-based $(\{1, k\}, n)$ hierarchical secret sharing scheme. SCIS 2018, 2A3-4 (2018) (in Japanese)

[58] Shima, K. and Doi, H.: A study on hierarchical secret sharing schemes applicable to any level based on XOR operations. IEICE Technical Report, Vol.117, No.487, 81–88 (2018) (in Japanese)

[59] Shima, K. and Doi, H.: XOR-Based Hierarchical Secret Sharing Scheme. IWSEC 2018, LNCS 11049, 206–223 (2018) **[Refereed]**

[60] Shima, K. and Doi, H.: A Hierarchical Secret Sharing Scheme Based on Information Dispersal Techniques. ICISC 2018, LNCS 11396, 217–232 (2018) **[Refereed]**

[61] Shima, K. and Doi, H.: My Interest of Study on the Hierarchical Management of Information Secrecy. IPSJ MAGAZINE, Vol.59, No.8, Reports: The 2017 IPSJ Best Paper Award, 723 (2018)