Received 27 June 2022, accepted 12 July 2022, date of publication 20 July 2022, date of current version 27 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3192645

THEORY

Simple Approach to Realizing Verifiable Secret Sharing for Secure Cloud System

KEIICHI IWAMURA[1], (Member, IEEE), AND
AHMAD AKMAL AMINUDDIN MOHD KAMAL[2], (Member, IEEE)
[1]Department of Electrical Engineering, Tokyo University of Science, Tokyo 125-8585, Japan
[2]Department of Information and Computer Technology, Tokyo University of Science, Tokyo 125-8585, Japan

Corresponding author: Ahmad Akmal Aminuddin Mohd Kamal (ahmad.amin@rs.tus.ac.jp)

ABSTRACT In (k, n) threshold secret sharing, a secret is converted into n shares such that any threshold k or more shares allow for the reconstruction of this secret; however, the total storage cost increases. By contrast, asymmetric secret sharing reduces the total shares to be stored. When implementing secret sharing in the cloud, if malicious players submit forged shares during the reconstruction process, the reconstructed value will differ from the original secret. Therefore, a method that quickly verifies the integrity of the restored secret should be developed. Many research papers investigate cheater detection/identification for (k, n) threshold secret sharing. However, most of them require additional information, such as an authenticator. Harn *et al.* proposed a method for cheat detection using only the shares for (k, n) threshold secret sharing. In this study, we improved and extended the method proposed by Harn *et al.* to realize the detection and identification of shares forgery (cheating) in asymmetric secret sharing suitable for a cloud system. The proposed method uses the shares generated during asymmetric secret sharing to reconstruct and verify the secret. We also included an attack that assumes a cloud system and shows that most methods cannot work against it. Finally, we discussed the requirements for a secret sharing scheme suitable for the cloud and showed that the proposed method is ideal for use in a cloud environment.

INDEX TERMS Secret sharing, verifiable secret sharing, probability of successful cheating.

I. INTRODUCTION

Secret sharing is one method for concealing and storing important data [1], [2]. A (k, n) threshold secret sharing is a method in which a single piece of secret information (input) is divided into n different values (known as shares) and stored in n different servers. The original secret information can be reconstructed from any k $(k \leq n)$ number of shares, where k is the threshold. However, any of the $k - 1$ or a smaller number of shares does not reveal secret information, thereby realizing information-theoretic security. The classic method for (k, n) threshold secret sharing is Shamir's (k, n) threshold secret sharing (hereinafter referred to as Shamir's (k, n) method) [1].

However, the problem with the secret sharing method is that one piece of secret information is converted into n shares; therefore, the total amount of data to be stored increases by n times. A ramp secret sharing that reduces the share size has been proposed [3]; however, the required number of servers remains constant at n. When considering the security of ramp secret sharing, the secret information may leak in stages according to the number of shares leaked. An asymmetric secret sharing method that reduces the number of shares to be stored (i.e., the number of servers required) has been proposed [4]. Because the asymmetric secret sharing method can reduce the number of servers to at most $k - 1$ servers, it can be configured with a minimum of only one server. However, the security achieved regarding the confidentiality of secret information is computational rather than information-theoretical.

Shamir's (k, n) method maintains the confidentiality of secret information even against the leakage of up to $k - 1$ shares. However, even if one false share is included during the reconstruction process, the correct result cannot be restored. There are many verifiable secrets sharing methods based on Shamir's (k, n) method that realize verification

The associate editor coordinating the review of this manuscript and approving it for publication was Sathish Kumar.

functionality to verify the correctness of the reconstructed result [5]–[26]. However, most of the proposed methods thus far often require additional data other than the original shares to verify the correctness of the reconstructed result, and the probability of successful cheating cannot be set to 0. Here, the cheating success rate refers to the probability that the adversary's cheating will succeed by chance, and malicious activity cannot be verified even if the shares have been tampered with.

An error correction code is one method that reduces the probability of cheating success to zero [13], [27]. However, this method requires many servers to store parity checks. Moreover, when an error (or false share) exists, the total computational cost is much higher than that of other conventional methods because of the error correction process. In addition, Harn and Lin proposed a verification method that uses only the original shares [19]; however, there are complex cases to be considered. In addition, although the probability of cheating success has not been described, it is assumed that the probability of cheating success cannot be set to zero, and the total computation cost also increases significantly as n and k increase.

However, the above verification methods only consider the security related to the reconstruction of one secret input and do not examine the security when many secret inputs are distributed and stored in the cloud system. This study describes another attack method for storing many secret inputs using the secret sharing method and shows that most conventional verification methods cannot cope with it.

Moreover, in this study, the asymmetric secret sharing method (which cannot verify the correctness of the restored value) is extended to realize verification functionality using only the original distributed shares. Furthermore, we set the requirements for a verifiable secret sharing method suitable for the cloud and show that the proposed method meets all the requirements well.

Conventional attack (cheating) methods can be classified into a CDV model (that assumes that the adversary knows the secret information) and an OKS model (that assumes that the adversary does not know the secret information). This study considers the OKS model and subsequently extends it to the CDV model and discusses its security.

A. OUR CONTRIBUTIONS

This study focuses on solving the problem of verifying the correctness of reconstructed secret information when using threshold secret sharing. We propose a simple process of verification using asymmetric secret sharing that can verify the correctness of the reconstructed result by performing several reconstructions. The proposed method minimizes the total amount of information needed to be stored compared to other conventional methods, including the original (k, n) threshold secret sharing method proposed by Shamir [1]. The contributions of this study can be summarized as follows.

- We propose a new verification method that extends the original asymmetric secret sharing. Moreover, in our proposed method, only the original shares generated are used to perform an efficient verification process without additional information, such as the authenticator data used in most common methods. This was implemented by performing multiple reconstructions using the shares to find any variations in the reconstructed results. We also include a detailed security analysis of the proposed method.
- We introduced the idea of a data replacement attack when considering an application in a cloud system where there is a lot of data being stored at once (by contrast, most conventional verification methods only considered a single secret/reconstructed information). We demonstrate that our proposed method is secure against this type of attack. Additionally, we discuss the security of conventional methods against this type of attack.
- We show that our proposed verification method using secret sharing can also be extended to be secure against the CDV model of cheating, where the adversary knows the actual secret information and tries to cheat the owner.
- In addition, we propose the requirements for an ideal verifiable secret sharing in a cloud system. We present a clear evaluation of our proposed method and other conventional methods against the set requirements and show that our method can fulfill the set requirements efficiently.
- Finally, we include a detailed comparison, including the computation cost and probability of cheating success for our proposed methods and conventional verifiable secret sharing methods. We show that our proposed method requires only t times of pseudorandom number generation and g times of reconstruction.

B. ORGANIZATIONS

The remainder of this study is organized as follows. Section II introduces the building blocks of our proposed method, and Section III explains related work. In Section IV, we present our new protocols for realizing verifiable secret sharing and discuss the security of our proposed method. In Section V, we discuss the effectiveness of our method and other conventional methods for the CDV model of cheating and security against clouds. Finally, in Section VI, we compare the proposed method with conventional methods, and clarify the characteristics (and advantages) of our method.

II. PRELIMINARIES

In this section, we introduce fundamental backgrounds and techniques used in the proposed method.

A. DEFINITIONS OF A VERIFIABLE SECRET SHARING

A secret sharing method that is capable of detecting cheating (or forging of shares) was first presented by Tompa and Woll [29], where $k - 1$ or less dishonest players submit "false" shares during the reconstruction process. The cheating will succeed if a player reconstructs an incorrect secret information. Verifiable secret sharing enables players to verify that their shares of a (k, n) threshold secret sharing are consistently generated by the dealer. To rephrase, without revealing the secret and the shares, players can verify that any subset of k or more than k shares define the same secret, but any subset of fewer than k shares cannot determine the same secret [16]. One of the first notions and objectives of verifiable secret sharing as defined by Benaloh as follows [28].

Notion 1: k-consistency:

A set of n shares is said to be consistent if any subset of k of the n shares defines the same secret.

In this study, we emphasize the specific problem of cheater detection and/or identification by the retriever for asymmetric secret sharing, such that there is only a small probability that any subsets of malicious servers can fabricate their shares to deceive the retriever during the reconstruction.

B. CDV AND OKS MODELS

In this study, we realize a verifiable asymmetric secret sharing system that can detect shares forgery by malicious adversaries. For example, suppose a scenario where colluding malicious players want to cheat another player by submitting forged shares during the reconstruction process so that the reconstructed value is different from the original secret information. Typically, such a cheating scenario can be modeled in two ways: CDV [15] and OKS [14] models. The main difference in both models lies in the knowledge of the cheaters, as shown below:

- CDV model: Carpentiari, De Santis, and Vaccaro (CDV) first considered a model in which cheaters who know the secret try to make another user reconstruct invalid secret information.
- OKS model: Ogata, Kurosawa, and Stinson (OKS) introduced another model assuming a weaker cheater who does not know the secret in forging their shares.

C. (k, n) THRESHOLD SECRET SHARING

Secret sharing is known as (k, n) threshold secret sharing when it satisfies the following conditions:

- Any $k - 1$ or fewer shares reveal no information about the original secret input s;
- Any k or more shares enable the reconstruction of the original secret input s.

The classic method for (k, n) threshold secret sharing is the Shamir (k, n) method [1]. In this method, all computations were performed in a finite field $GF(p)$.

Shamir's (k, n) method uses the following protocols for the distribution and reconstruction of secret input s.

Protocol 1: Distribution of secret input s

1) Selects any prime number p such that $s < p$ and $n < p$.
2) The dealer selects $k - 1$ random numbers a_i $(i = 1, \ldots, k-1)$ from $GF(p)$ and generates a random polynomial $f(x)$ as follows:
$$f(x) = s + a_1 x + \cdots + a_{k-1}x^{k-1} \tag{1}$$
3) The dealer inserts the ID x_j $(j = 1, \ldots, n)$ of each server into x in Equation (1), calculates the shares $f(x_j) = W_j$ corresponding to each ID, and sends them to all servers.

Protocol 2: Reconstruction of secret input s

1) The player who wants to restore secret input s collects any k shares and their pair of IDs. Let us assume that the shares are W_h $(h = 1, \ldots, k)$ and its corresponding IDs are x_h.
2) Substituting x_h and W_h into Equation (1) and solving k simultaneous equations to obtain the secret input s.

From the above, the computed shares become a point on the curve of the polynomial function $f(x)$, and the reconstruction of the secret input is a process of reproducing Equation (1) from the collected k points.

D. ASYMMETRIC SECRET SHARING

In the (k, n) threshold secret sharing method, the total shares generated cannot be made smaller than the original secret information (there is a problem in which the storage capacity efficiency is poor). A (k, L, n) ramp secret sharing was proposed to solve this problem, where the data size of the share is reduced by $1/L$ [3]. There was a problem in which the secret information could leak gradually even when shares were below the threshold k. Therefore, instead of reducing the data size of the share itself, asymmetric secret sharing was proposed to reduce the total number of shares held by computing servers [4].

In the original asymmetric secret sharing [4], t servers are selected from n servers as *key servers* (by contrast, the proposed method can eliminate the need for any key servers). Because these key servers do not hold any shares regarding the secret information but only the information (*key*) for generating pseudo-random numbers, the total number of shares can be reduced. Each key server generates a pseudo-random number in response to the user's request, and the user determines the distribution function by using the generated pseudo-random numbers as shares. The shares possessed by the remaining servers are then calculated using the distribution function determined. A server that stores shares regarding secret information is called a *data server.*

In addition, $ID[y]$ $(y = 1, \ldots, r)$ is assigned to each user who distributes his/her own secret information for user identification and $dID[s_{ij}]$ $(i = 1, \ldots, m)$ is assigned to each of the m secret information $s_{1j}, \ldots, s_{mj}$ $(j = 1, \ldots, r)$ possessed by each user for data identification.

The construction of asymmetric secret sharing is as follows. First, the following distribution function $f(x)$ is generated for each secret information s_{ij} $(i = 1, \ldots, m, j = 1, \ldots, r)$. Here, all computations were performed in $GF(p)$.
$$f(x) = s_{ij} + a_{i1}x + a_{i2}x^2 + \cdots + a_{ik-1}x^{k-2} \tag{2}$$

Each server sends the value $f(x_j) = W_{ij}$ obtained when each server identifier x_j is assigned to the aforementioned $f(x)$ as a shared W_{ij}. In addition, the data identification $dID[s_{ij}]$ assigned to each secret information must be smaller than the data size of the secret information, and the following relation holds between $dID[s_{ij}]$ and s_{ij}. Here, $H(A|B)$ represents the entropy related to A when the information B is known.

$$H(s_{ij}) = H\left(s_{ij}|ID[s_{ij}]\right) \tag{3}$$

The user then sends the user identifier $ID[y]$ $(y = 1, \ldots, r)$ to the key server, and using the key key_j that the key server has and the received $ID[y]$, the key server computes the following and sends it to the user: Here, it is assumed that $Enc(a, b)$ represents the process of encrypting a by using key b.

$$Eid(y, j) = Enc(ID[y], key_j) \ \ (j = 1, \ldots, t) \tag{4}$$

The user then uses this information to encrypt their data identifier $dID[s_{ij}]$ and generates the encryption result $q_{ij} = Enc(dID[s_{ij}], Eid(y, j))$ for all m secret pieces of information. The coefficients $a_{i1}, \ldots, a_{ik-1}$ in the distribution function for each secret information s_i $(i = 1, \ldots, m)$ are defined as $W_{1j} = q_{1j}, W_{2j} = q_{2j}, \ldots, W_{mj} = q_{mj}$, so that they correspond to the information of the key server. We also include the detailed protocols for the original asymmetric secret sharing in Appendix.

III. PREVIOUS WORK FOR VSS

In this section, we review the known works for realizing verifiable secret sharing, in particular, detection and identification of malicious actions. Some methods for realizing verifiable secret sharing include the process of using additional data called an authenticator that has a specific relationship with the secret input (hereinafter referred to as the authenticator method) [5]–[12], the method of detecting and correcting false shares (or errors) using Reed-Solomon code (referred to as the RS method) [13], and a method that can verify the reconstructed result using only the distributed shares (for example, Harn *et al.*'s method [19]), etc.

The details of the typical authenticator method, RS method, and Harn *et al.*'s method are shown below.

A. AUTHENTICATOR METHOD THAT EXPONENTIATES THE SECRET INPUTS

Using $a = s^2$, which is the square of the secret input s, as an authenticator, s and a are then secret-shared; if the reconstructed result satisfies the condition of $a = s^2$, we can assume that there is no cheating (or false shares) [6]. However, the correctness of the secret input cannot be verified using $GF(2^m)$. If the adversary tampers with the shares of both the secret information and authenticator, both false secret input $s' = s + \triangle s$ and false authenticator $a' = a + \triangle a$ are restored.

At this time, the adversary must set $a' = (s')^2$ for the cheating to succeed. However, in $GF(2^m)$, the adversary can generate an error such that $\triangle a = 2s \triangle s + (\triangle s)^2 = (\triangle s)^2$; therefore, the probability of cheating success is one. Therefore, there is a limitation that $GF(2^m)$ is not used.

In addition, even if $GF(2^m)$ is not used, the reconstructed result using a false share may occur as $a' = (s')^2$; therefore, the probability of cheating success is $1/p$. Moreover, because the shares for the authenticator are stored in addition to the shares of the secret information, each server's total required storage cost is doubled. Similarly, the method with improved resistance to $GF(2^m)$ also has the limitation that no finite field can be used because the adversary's cheating success rate is 1 when $GF(3^m)$ is used [7].

B. AUTHENTICATOR METHOD THAT DECOMPOSES SECRET INFORMATION INTO BIT STRINGS

The method in [8] is specialized for verification in $GF(2^{2m})$. The secret information s is first decomposed into half, the bit string $s = (s_1, s_2)$ is generated, and the secret information s and authenticator $a = s_1 s_2$ are secret-shared. Only if the reconstructed secret information s' and authenticator a' fulfilled the following condition, the reconstructed result was considered valid.

$$a' = s'_1 s'_2 \tag{5}$$

This method does not adapt to any bit string. Method [8] is valid only when the bit length is even and the secret information $s \in GF(2^{2m})$. However, the method proposed in [5] also decomposes the secret information into N bits, but it is valid only when $s \in GF(2^{Nm})$ and an arbitrary field cannot be set. In addition, the required storage cost increases because the shares of the authenticator are also saved. In addition, the probability of successful cheating was $1/p$.

From the above, the authenticator method can verify the reconstructed result efficiently by simply reconstructing and comparing the secret information and the authenticator. But, very few methods can identify dishonest servers. However, the authenticator method has no restrictions on the parameters n, k, and in many cases, $n \geq k$ can be realized.

C. VERIFIABLE SECRET SHARING BASED ON REED-SOLOMON (RS) CODE

Shamir's (k, n) method can be constructed using the RS code [13], [27]. Because this method can detect all cheating within the set error range, the cheating success probability can be set to 0, identifying dishonest servers. However, if the secret input is saved as one share, it will be leaked simply by attacking the server. Therefore, the secret input is encrypted by a linear combination of $k - 1$ random numbers to form an information sequence.

When the error correction capability for the false share of the e symbol is given, $2e$ parity symbols are attached to it. Therefore, the number of shares is more significant than other methods, and many computing servers are required. In addition, the error correction code is performed during the distribution process, and if the syndrome during reconstruction is not zero, an error correction process is required; therefore, the total computation cost is much higher than that of the other methods and is not efficient.

D. HARN ET AL.'s METHOD

Consider $j > k$ shares out of n shares. In this case, there is a set of shares $u =_j C_k$, and u values can be reconstructed. If the u restored values match, the j distributed shares are valid. In other words, it is recognized that the selected j shares are on the same line as the polynomial function. By contrast, if the reconstructed values do not match, it is assumed that the j shares contain invalid shares. To rephrase, a share that does not exist on the same line is a false share.

Therefore, in Harn *et al.*'s method [19], reconstruction is performed for all combinations of $n \geq j \geq k$ shares, and matching of the reconstructed values is verified. If servers corresponding to j shares with the same reconstructed value are found, then the values for the remaining servers are individually restored to identify the dishonest server. In the following, the number of dishonest servers or adversaries is c. Additionally, we consider the following three adversaries separately.

Adversary 1: Non-colluding adversary

The presence or absence of false shares can be detected when $j \geq k+1$, and the dishonest servers can be identified when $j - c > k$.

Adversary 2: c adversaries collude; however, the shares are output simultaneously and cannot be changed.

The presence or absence of false shares can be detected when $((c < k) \cap (j \geq k+1)) \cup ((c \geq k) \cap (j - c \geq k))$, and dishonest servers can be identified when:

$$((c < k) \cap (j - c \geq k+1)) \cup ((c \geq k) \cap (j - c > c+k-1))$$

Adversary 3: c adversaries collude but can change the output shares after seeing the output from an honest server.

The presence or absence of false shares can be detected when $j - c \geq k$, and the dishonest servers can be identified when $(j \geq k+1) \cap (j - c > c + k - 1)$.

From the above, Harn *et al.*'s method does not include $n = k$, and complicated case classification is performed on the premise that $n > k$. Moreover, cheating and dishonest servers can be identified only if the conditions in each case are satisfied. In addition, because reconstructions are performed ${}_jC_k$ times, the computational cost becomes enormous as j and k become large.

IV. PROPOSED METHOD

References [5]–[8] demonstrate that cheating could be detected by using additional information (e.g., authenticator). However, the required storage cost of each server will increase due to the additional information needed. However, Harn *et al.* demonstrated that cheating could be detected and identified by using only the original shares generated by Shamir's (k, n) method. However, as mentioned previously, the inability to perform detection and identification of cheaters when each condition is not fully satisfied is the greatest disadvantage of the Harn *et al.* method. Moreover, the distribution of input using Shamir's (k, n) method in [20], where n shares are generated for each input, means that the total data saved increases.

In this study, we enhance the Harn *et al.* method to realize the detection and identification of cheaters efficiently without using any additional information, such as authenticators used in [5], [6], and [7]. To achieve this, instead of Shamir's (k, n) method, we implement asymmetric secret sharing in [4] and reduce the total number of shares to be stored, thus reducing the number of servers required. However, in the proposed method, we further improve the secret sharing method in [4] such that the role of key servers is realized by the owner, further reducing the number of servers required. In addition, we improved the cheating detection process such that detection and identification of cheaters are possible with fewer conditions needing to be fulfilled, thus producing a better result than in Harn *et al.*'s method. A detailed comparison of computational cost and advantages/disadvantages of each method is presented in Section V.

A. PROPOSED ALGORITHM

The Shamir (k, n) method is used to share secret input. In addition, n server IDs $x_1, \ldots, x_n$ are made available to the public. However, all IDs are different and do not include 0. In addition, $n > k$, and all computations are performed using the prime number p. Furthermore, we consider an OKS model that assumes that the adversary does not know the secret input. Moreover, any data that can be represented by a sequence of bits can be transferred in the proposed method. For example, a number, a character in a document, a pixel in an image, etc.

In typical (k, n) threshold secret sharing, n servers are treated equally. However, in asymmetric secret sharing, n servers are divided into t key-servers that manage keys and $g = n - t$ data-servers that store shares. In the proposed method, the keys of the server are managed by the owner of the secret input. In this case, the owner can generate t keys at once by using a pseudo-random number generator with one initial value. Therefore, for transferring one data, the owner needs to remember one initial value (for generating t keys) and one key (for generating pseudo-random numbers in Step 1 of Protocol 3).

Therefore, n servers are not treated equally, and *there are virtually no key servers because the owner can manage the keys securely on their PC or smartphone*. Protocol 3 is the same process as asymmetric secret sharing, and for ease of understanding, we assume $t = k - 1$. In addition, the number of data servers is assumed to be $2 \leq g \leq k - 1$. Furthermore, considering that a large number of secret inputs are secret-shared and stored in the cloud, the information that identifies m secret inputs is expressed as $ID[s_i]$ $(i = 1, \ldots, m)$ (we will explain this in detail below).

However, $H(s_i) = H(s_i|ID[s_i])$ and $ID[s_i]$ were determined independently of s_i.

Protocol 3: Distribution Process

1) The owner has t keys key_j $(j = 1, \ldots, t)$ corresponding to t key servers and encrypts the secret input identifier $ID[s_i]$ $(i = 1, \ldots, m)$. Here, t number of pseudo-random numbers q_{ij} is generated for each i. However, $Enc(x, y)$ implies that x is encrypted with y.

$$q_{ij} = Enc(ID[s_i], key_j) \tag{6}$$

2) The owner uses the pseudo-random number sequence $Q = [q_{i1}, \ldots, q_{it}]^T$ generated in Step 1 and the following ID sequence of the key-servers, and computes the coefficient vector $A(i) = [a_{i1}, \ldots, a_{ik-1}]^T$ of Equation (1) from the following Equation (8).

$$X' = \begin{bmatrix} x_1 & \cdots & x_1^{k-1} \\ \vdots & \ddots & \vdots \\ x_t & \cdots & x_t^{k-1} \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} q_{i1} \\ \vdots \\ q_{it} \end{bmatrix} = \begin{bmatrix} s_i \\ \vdots \\ s_i \end{bmatrix} + \begin{bmatrix} x_1 & \cdots & x_1^{k-1} \\ \vdots & \ddots & \vdots \\ x_t & \cdots & x_t^{k-1} \end{bmatrix} \begin{bmatrix} a_{i1} \\ \vdots \\ a_{ik-1} \end{bmatrix} \tag{8}$$

3) The owner uses the coefficient vector generated in Step 2 and computes shares $W_{it+1}, \ldots, W_{in}$ for data servers $x_{t+1}, \ldots, x_n$ using the same procedure as in the (k, n) threshold secret sharing method. It is then sent to each data server together with $ID[s_i]$.
4) The data servers store the $ID[s_i]$ and W_{ij} in association with each other.

Protocol 4: Reconstruction Process

1) The owner sends the $ID[s_i]$ of the secret input that they want to reconstruct to all data servers and use his/her key key_j to generate t pseudo-random numbers $q_{ij} = Enc(ID[s_i], key_j)$.
2) Data servers that receive the $ID[s_i]$ sends the corresponding shares W_{ij} to the owner.
3) Regarding the generated t pseudo-random numbers q_{ij}, the owner selects shares from g data servers individually and reconstructs the secret input. This reconstruction was repeated g times.
4) If two or more reconstructed values match, the owner assumes that the data servers used for the reconstruction are honest and adopts the reconstructed result.
5) The owner identifies servers that do not match the restored values as malicious servers. When all the reconstructed values do not match, they are not adopted because of an unknown error.

B. SECURITY OF THE PROPOSED METHOD

The security of the asymmetric secret sharing method is that as long as

1) there are only $k - 1$ or fewer data servers,
2) the owner manages the key securely, and
3) the adversary has no information on the pseudo-random number (=shares) generated from the key,

the adversary will not be able to obtain any secret information even if they attack all the data servers. In addition, in conventional methods (including Shamir's (k, n) method), secret information is leaked if k shares are collected owing to server leakage, even if the owner does not allow it. However, in the asymmetric secret sharing method, if the owner manages its key securely, secret information will not be leaked, even if the information of all data servers is leaked. However, the asymmetric secret sharing method will fail if the adversary can learn the generated pseudo-random number even without knowing the key. The asymmetric secret sharing method only realizes computational security that depends on the pseudo-random number generation method used.

The security for realizing *completeness* is as follows. The prime p is assumed to be sufficiently large. As with confidentiality, if the owner keeps the key used secure, $t = k-1$ shares generated from that key are always correct and unknown to the adversary. Therefore, if the reconstruction process is performed while changing the shares from the data server one by one, the detection of malicious activity is possible, except in the case of the following accidental match.

For example, if the false shares output by two data servers lies on the same straight line, even if they are combined with other $k - 1$ valid shares, malicious cannot be detected. Here, the second "false" share happens to be on the invalid line restored using the first "false" share, and the probability of successful cheating by the adversary is $1/p$. However, when there are three data servers, the third share must also be on the same invalid curve for cheating to succeed, and the cheating success probability drops to $1/p^2$. Therefore, if the number of data servers is g, then the probability of cheating success is $1/p^{g-1}$.

The asymmetric secret sharing method can also be applied to $t \leq k-1$ (it suffices to pre-determine $k-1-t$ number of a_{ij} in the coefficient vector $A(i)$). Therefore, when $t = k-2$, with t correct shares, the third "false" share must be on the same invalid curve generated from the first two "false" shares; therefore, the probability of cheating success is $1/p^{g-2}$.

Therefore, regarding the completeness of our proposed method, the probability that the adversary can cheat successfully can be minimized by setting the number of key servers to $t = k - 1$ and data servers to $g = n - t = k - 1 \geq 2$. However, since $k - 1 \geq 2$, parameter k needs to be three or more, and $k = 2$ cannot be selected.

Here the original asymmetric secret sharing realizes only the confidentiality of secret information because the secret information can be reconstructed using only one data server by selecting $k = 2$. However, parameter k and the number of servers will increase when considering compatibility with completeness. We discuss this further in Section V.

If two or more reconstructed values match during the g times of reconstructions, i.e., if two data servers are honest in the proposed method, malicious servers can be identified.

V. DISCUSSION AND CONSIDERATION

A. SECURITY AGAINST CDV MODEL OF CHEATING

The algorithm presented in Section IV is also valid for the CDV cheating model. However, the number of data servers g must be $2 \leq g \leq k-2$. Therefore, parameter k must be more than four.

In the CDV model, the adversary knows the secret information. If the owner who wishes to reconstruct their secret information (the *restorer*) can be made to get a "false" reconstructed result, the attack is considered successful. Therefore, in the case of $g = k-1 \geq 2$ shown in Section IV, because the adversary can reproduce equation (1) (hereinafter referred to as the distribution polynomial) using $k-1$ shares from data servers and secret information, the adversary will also be able to learn about all the shares given by the owner.

In other words, if the adversary constructs an invalid curve by combining the shares output by the owner and one "false" share and sets the other "false" shares to also follow the invalid curve, no matter which data server is used, the reconstructed "false" result will always match, and the attack will succeed.

However, if the number of data servers g is set such that $2 \leq g \leq k-2$, the adversary cannot reproduce the distribution polynomial even if they know all the shares stored in the data servers in addition to the original secret information. Therefore, it is impossible to know all the shares computed by the owner. Thus, the attack fails. However, it has a probability of cheating success of $1/p^{g-1}$ because of a coincidence match.

However, consider the security of other conventional methods against the CDV model of cheating. In the authenticator method, if the adversary can know the shares of $k-1$ servers, the distribution polynomial can be reproduced from $k-1$ shares and secret information. Therefore, the remaining shares held by $n-k+1$ honest servers can also be known. Thus, if $k-1$ malicious servers are made to hold "false" shares that can be verified against a valid share, the attack will succeed. By contrast, the asymmetric secret sharing method can manipulate the number of data servers used. However, the authenticator method cannot prevent this because the number of servers is $n \geq k$, and n cannot be set to be less than k.

In the RS method, the distribution polynomial can be reproduced from $k-1$ shares and secret information. The parity is also known; however, it is challenging to find a code with the same parity for different information sequences. In conclusion the RS method is effective against the CDV model.

The effectiveness of Harn *et al.*'s method against the CDV model will be discussed later.

B. ATTACKS ON CLOUD STORAGE OF LARGE AMOUNTS OF SECRET INFORMATION

Consider the case in which the owner secret-shared and stores multiple secret information in cloud storage. At this time, the $ID[s_i]$ that identifies the secret information is associated with the computed shares and saved. Because $ID[s_i]$ specifies that the secret information that the owner wants to reconstruct during the reconstruction process does not contain the actual secret information, the owner can send it without any encryption.

However, the adversary may replace it with a different $ID[s'_i]$ instead and sends it to the cloud. Alternatively, even if $ID[s_i]$ is encrypted, the adversary can observe past communication and replace the encrypted $ID[s_i]$ with different encrypted information $ID[s'_i]$ before sending it to the cloud. Moreover, we could also consider the case where the adversary is already part of the cloud system and tells the server an $ID[s'_i]$ that is different from the $ID[s_i]$ sent by the owner. Therefore, to realize completeness, it is also necessary to consider security against data replacement and the falsification of shares.

Here, the server sends shares for $ID[s'_i]$ to the owner. In the authenticator method, when the shares are collected and reconstructed, the restored value is adopted if the relationship between the secret information and the authenticator matches correctly. However, if the owner specifies the reconstructed secret information, the attack is considered successful. In RS and Harn *et al.*'s methods, a reconstructed result that passes the verification process can be obtained; however, the attack still succeeds because it may not be the secret information specified by the owner.

As aforementioned, if incorrect secret information that the owner does not specify can be transmitted to the server, the adversary can get the owner to obtain incorrect secret information without tampering with the shares of $k-1$ servers. In conventional research on verifiable secret sharing methods, because only one secret information is assumed, an attack that replaces the information that specifies the reconstruction for multiple secret information is not expected, and all attacks succeed. Hereafter, this attack is called *data replacement attack*.

In the proposed method, the data server sends the shares for $ID[s'_i]$, but the owner believes that it is the shares for the $ID[s_i]$ that they had requested and reconstructed it. An adversary can get the owner to obtain incorrect reconstructed secret information using the original asymmetric secret-sharing method, which realizes only the confidentiality of the secret information. However, when our proposed method is used, the shares sent for the $k-1$ shares generated by the owner are shares of other secret information. Therefore, the reconstructed results were different. The reconstructed results may also not match if the data servers fail. Thus, the proposed method could identify that there is a malicious action, but it cannot differentiate whether the action is caused by a malicious administrator or broken data server. However, there is still a $1/p$ probability for cheating to succeed in which the two restored values match.

The argument above shows that the owner may not obtain the correct reconstructed result using our proposed method; however, the owner will never obtain an incorrect reconstructed result as a valid result, and the data replacement attack fails.

This attack can also be dealt with by using the secret information linked to the $ID[s_i]$ as the secret information and checking the ID and $ID[s_i]$ during the reconstruction process. However, the amount of data is expected to increase further.

In general, the person who can access the data stored in the cloud is often limited to the owner, as shown in Section V, and the owner confirms that the reconstructed value is correct (including when there is server failure) by using verifiable secret sharing. Therefore, when the servers are managed correctly, it is unlikely that all the reconstructed results will be different. The conventional methods cannot detect the data replacement attack, but the proposed method can; therefore, it is possible to request an investigation from an organization that manages the cloud servers.

C. CHEATING SUCCESS RATE OF HARN ET AL's METHOD

In the case of Adversary 1, regarding the line created from $t \geq j - c$ number of valid shares and k out of c number of "false" shares, if the shares of the remaining $j - k$ servers also lie on the same line, the reconstructed value will be the same for any k of the j servers and no malicious activity will be detected.

Here, if $t < k$, a "false" share is included, and the obtained line is different from the actual distribution polynomial function. Because the correct line is restored only if condition $t \geq k$ is satisfied, shares that do not lie on the line can be regarded as "false" shares. However, for cheating detection only, there is no condition of $t \geq k$, so even if c adversaries do not collude, malice is not detected if the remaining $j-k$ shares happen to be on the same "false" curve. Therefore, in the case of Adversary 1, cheating succeeds with a probability of $1/p^{j-k}$.

For identifying malicious servers in Adversary 1, it is assumed that the number of honest servers is $k + 1$ or more, so there are always j servers that match all combinations. However, if $c > k$, all combinations can be set to match by chance, even between dishonest servers, so it is impossible to determine which is correct, and malicious servers cannot be identified. Therefore, it can be said that the security of Harn *et al.*'s method has not been strictly evaluated.

For Adversary 2, in the case of $((c < k) \cap (j \geq k + 1))$, the same argument as for Adversary 1 holds. However, for the other condition $((c \geq k) \cap (j - c \geq k))$ of Adversary 2, because $c \geq k$, the adversary can reproduce the distribution polynomial function from the shares stored in the server and know the secret information, and can also know the shares of honest servers. Therefore, this is the case with the CDV model.

However, because $j - c \geq k$, that is, there are k or more honest servers, the correct distribution polynomial function is always restored. If $c > k$ is set from $c \geq k$, "false" reconstructed results can be matched even in dishonest servers; however, because different reconstructed results exist, cheating detection can be performed. Therefore, Harn *et al.*'s method can handle the case where $j - c \geq k$ for the CDV model. However, this condition is not always satisfied. On the other hand, by setting parameter g to be $2 \leq g \leq k - 2$ in our proposed method, cheating can always be detected even in the CDV model, except for the cheating success probability of $1/p^{g-1}$.

In addition, the condition $((c < k) \cap (j - c \geq k + 1)) \cup ((c \geq k) \cap (j - c > c + k - 1))$ is applied to identify malicious servers in Adversary 2. Only one legitimate set of servers matches the restored values in the former. By contrast, dishonest servers can be identified in the latter because the number of valid reconstructed results is greater than that of the invalid reconstructed results.

Because the asymmetric secret sharing method does not inform the adversary of the shares generated by the owner, the situation of Adversary 3 does not exist. Therefore, the evaluation of Adversary 3 is omitted.

D. COMPARISON OF VERIFIABLE SECRET SHARING METHODS FOR CLOUD SYSTEM

The following system is assumed when considering the security management of secret information using verifiable secret sharing in the cloud system:

- The system is composed of a cloud system consisting of multiple computing servers (and its administrator) and an owner who deposits the secret information in the cloud.

The owner registers as a user before using the cloud system. The cloud then allocates the required number of servers to the owner, whose registration is accepted, and allows the shares of secret information to be stored. It is also assumed that the owner maintains the shares of multiple secret information, but the access is limited (controlled) such that only the owner can access the stored data.

The service charge of the system increases depending on the amount of data to be stored and the number of servers used by the owner. Alternatively, the amount of data that can be stored and the number of servers that the owner can use are limited. We also assume that the adversary is inside the system and knows secret information.

The following can be considered the requirements for verifiable secret sharing for the cloud system mentioned above.

Requirements:

1) The amount of data to be saved and the number of servers used were small.
2) Resistant to various attacks such as CDV model and data exchange attack.
3) It is possible to verify whether the reconstructed result is correct efficiently.
4) Malicious servers can be identified.
5) The probability of cheating success can be reduced.

Table 1 presents a comparison of the methods in the above system. In Table 1, **Case1** is the case of cheating verification only, and **Case2** includes identifying the malicious servers. In addition, prime p' is a relatively small value that does not depend on the probability of cheating success, prime p is a sufficiently large value, and $|x|$ represents the number of bits of x. Let e in the RS method be $k - 1$. In addition, $\times$ indicates that it cannot be dealt with and $\triangle$ implies that there is a probability of cheating success.

Notations: A–E in Table 1 represent the following:

- A: Computation cost for generating a share.
- B: Computation cost of the error correction code.
- C: Computation cost for generating one pseudo-random number

TABLE 1. Comparison with conventional method.

	Proposed method	**Authenticator method**	**RS method**	**Harn et al.'s method**
Required number of servers	$1 < g \le k-2$	$n \ge k$	$n \ge 3k-2$	$n > k$
Amount of data to be stored	$g\|p\|$	$2n\|p\|$	$2n\|p'\|$	$n\|p\|$
CDV model of cheating	△	×	○	With condition
Data replacement attack	△	×	×	×
Computation cost (distribution)	$tC + gD$	$2nA$	$kA + B$	nA
Computation cost (Case1)	$tC + gD$	$2D$	$2(k-1)D$	${}_jC_kD$
Additional computation cost (Case2)	0	×	E	$(n-j)D$
Probability of cheating success	$1/p^{g-1}$	$1/p$	0	$1/p^{j-k}$

- D: Computation cost for reconstructing secret information.
- E: Computation cost for the error correction process

The following can be inferred from Table 1. For the minimum number of servers (hereinafter referred to as the *minimum server setting*), if parameter k of our proposed method is $k = 4$ and the others are $k = 2$, the minimum number of servers is two for the authenticator method, four for the RS method, three for Harn *et al.*'s method, and two for our proposed method.

In this case, the proposed method does not leak any secret information, even if both servers are attacked. Therefore, for other methods to resist the attack of up to two servers, if $k = 3$, the minimum number of servers is three for the authenticator method, seven for the RS method, and four for Harn *et al.*'s method. By contrast, the proposed method remained at two. This is referred to as the *same attack setting*. In addition, when the parameter k of all methods is $k = 4$, the minimum number of servers is four for the authenticator method, ten for the RS method, and five for Harn *et al.*'s method.

In addition, the minimum amount of data to be stored is $(3k-2)|p'|$ bits for the RS method and $(k-2)|p|$ bits for the proposed method. Therefore, if the number of bits of p' is $(k-2)/(3k-2)$ or less (approximately $1/3$) against the number of bits of p, the RS method is the smallest, followed by the proposed method. We can state that our proposed method satisfies requirement (1) from the above. However, the RS method does not meet requirement (1) regarding the number of servers.

Regarding support against the CDV model of cheating, the authenticator method cannot be used regardless of k, and Harn *et al.*'s method cannot be used because the conditions are not met regardless of k if the number of servers is at a minimum $k + 1$ and the two servers are malicious. The RS method is resistant to attacks on up to $k - 1$ servers, and the proposed method is resistant to attacks on all data servers.

Here, assuming a CDV model, the attacker knows the secret information, but if the attacker is already part of the system (for example, if the cloud administrator is the adversary), the adversary knows the secret information in other methods, and a CDV model can be realized. However, the proposed method does not leak any secret information even if the attacker is inside the system. Therefore, concerning our proposed method, the CDV model is effective only when the adversary learns secret information using another method. Furthermore, because our proposed method is resistant to data replacement attacks, we can state that it satisfies requirement (2) more strongly than other methods. However, the authenticator method cannot be used efficiently, except for verification in the OKS model.

In addition, the relationship between A to E is considered to be $A < D < B < C \ll E$. In addition, Step 2 in Protocol 3 (distribution process) of the proposed method can be realized by Lagrange interpolation. Therefore, the proposed method requires the most significant computation during the distribution process. However, the distribution can be preprocessed beforehand and is not crucial because it is performed only once.

In **Case1**, the RS method only needs to calculate $2(k-1)$ syndromes; therefore, in the minimum server setting, only a minimum of two reconstructions are required (except for Harn *et al.*'s method). Harn *et al.*'s method requires three reconstructions, even when the minimum servers $j = k+1$ are set. However, for ease of comparison, the computational cost of syndrome generation is assumed to be the same as the computational cost of Lagrange interpolation.

The number of reconstructions in the same attack setting increases to four for the RS method and Harn *et al.*'s method but remains at two for the authenticator and our proposed methods. However, in our proposed method, parameter k is larger than in other methods, and the computation cost for Lagrange interpolation is also high; however, the number of reconstructions is two at $k = 4$ in our proposed method and four times at $k = 3$ in the RS method and Harn *et al.*'s methods. Therefore, the proposed method is considered more efficient in terms of the computation required for reconstruction.

In addition, when parameter k increases, the number of restorations in the RS method and Harn *et al.*'s method also increases; however, the proposed method does not change because the number of data servers can remain at two. However, the cheating success probability, described later, remains at $1/p$. Because our proposed method requires the generation of t pseudo-random numbers, it is considered to have the highest overall computational cost. However, other methods require communication for t, and the communication process often takes longer than the actual computational process. Therefore, the proposed method realizes efficient processing as a whole and satisfies requirement (3) when communication is considered.

Corresponding to **Case2** is possible for all methods other than the authenticator method; however, the RS method has the highest additional computational cost. In addition, because the RS method can identify only up to $k-1$ malicious servers, only one of four malicious servers can be identified with the minimum server setting, and only two of seven malicious servers can be identified with the same attack setting. In addition, in Harn *et al.*'s method, malicious servers can only be specified when the conditions are met. For example, if one server is malicious, Harn *et al.*'s method cannot detect it when setting the minimum number of servers, but it can be detected by setting the number of servers to four.

In the proposed method, if $k = 5$ and the number of data servers is three, one malicious server can be identified. In addition, Harn *et al.*'s method requires more honest servers depending on k, and the computation required increases. However, our proposed method does not require an additional computation cost for **Case2** and can satisfy requirement (4) if there are two honest data servers, regardless of k.

Finally, the cheating success probability is fixed at $1/p$ in the authenticator method, and the cheating success probability can be reduced to 0 for up to $k-1$ malicious servers in the RS method. Harn *et al.*'s method and our proposed method are $1/p$ when assuming the minimum server setting; however, if the number of servers is increased by one, as described above, it becomes $1/p^2$.

The RS method is the best from the viewpoint of the probability of cheating success, but the RS method requires a more significant amount of computation than other methods. In our proposed method, the probability of successful cheating cannot be set to zero but can be arbitrarily reduced by selecting the number of data servers, and it can be said that requirement (5) is also satisfied.

Based on the above, the proposed method is considered a verifiable secret sharing method that is more suitable for the cloud than conventional methods (each with its own advantages and disadvantages), efficiently satisfying all the aforementioned requirements. In addition, these characteristics become more apparent as k and n increase.

VI. CONCLUSION

This study extended the asymmetric secret sharing method by adding a simple process to realize the verification functionality for the reconstructed result. The proposed method meets all requirements for the cloud ecosystem and realizes verification in a well-balanced and efficient manner.

We anticipate that implementation with a secure computation method can be performed successfully in a future study.

APPENDIX

ALGORITHM FOR ASYMMETRIC SECRET SHARING

Here, we present the detailed distribution and reconstruction protocols for the original asymmetric secret sharing proposed by Takahashi *et al.* [4] The number of key servers is determined at the time of cloud system configuration and is set to l units ($2 \leq l \leq k$). In addition, each coefficient in the distribution function of each secret information is expressed as $A(i) = [s_{ij}, a_{i1}, \ldots, a_{ik-1}]^T$ using a vector of degree k.

Protocol A.1: Distribution process

1) The user sends his $ID[y]$ ($y = 1, \ldots, r$) to the key servers $x_1, \ldots, x_l$.
2) Each key server that receives the $ID[y]$ uses its own encryption device and key key_j to compute $Eid(y, j) = Enc(ID[y], key_j)$ ($j = 1, \ldots, l$) and sends it to the user.
3) Upon receiving this information, the user computes the following pseudo-random numbers using the data identifier $dID[s_{ij}]$ ($i = 1, \ldots, m$) related to his/her secret information.

$$q_{ij} = Enc\left(dID[s_{ij}], Eid(y, j)\right)$$

4) The user first sets the $k-1-l$ degree of the partial vector $A'_{k-1-l}(i) = [a_{il+1}, \ldots, a_{ik-1}]^T$ in the coefficient vector $A(i) = [s_{ij}, a_{i1}, \ldots, a_{ik-1}]^T$ of k degree using true random numbers. Then, the remaining partial vector $A'_l(i) = [a_1, \ldots, a_l]^T$ in $A(i)$ is computed using the following equation (in addition to the pseudo-random numbers $Q = [q_{ij}, \ldots, q_{mj}]^T$ generated in Step 3 and ID sequence of the key servers):

$$ID\ of\ key\ servers : X' = \begin{bmatrix} x_1 & \cdots & x_1^{k-1} \\ \vdots & \ddots & \vdots \\ x_t & \cdots & x_t^{k-1} \end{bmatrix}$$

$$A'_l(i) = X'^{-1} Q$$

5) This allows the user to determine the partial vector $A(i)_{k-1} = [a_{i1}, \ldots, a_{ik-1}]^T$ of degree $k-1$ in the coefficient vector $A(i) = [s_{ij}, a_{i1}, \ldots, a_{ik-1}]^T$ for the distribution function of k degree.
6) In addition, the user calculates the shares $W_{il+1}, \ldots, W_{in}$ for data servers $x_{l+1}, \ldots, x_n$ using the same procedure as the (k, n) threshold secret sharing method based on the coefficient matrix generated in Step 4.
7) The user sends the generated shares $W_{1j}, \ldots, W_{mj}$ ($j = l+1, \ldots, n$) to each data server.

Protocol A.2: Reconstruction process

1) The user who wants to restore secret information s_i selects any k servers from n servers $x_1, \ldots, x_n$, and sends his $ID[y]$ and data identifier $dID[s_{ij}]$ of secret information s_i.
2) Key servers that receive $\left(ID[y], dID[s_{ij}]\right)$ generate $Eid(y, j)$ using its own key key_j and pseudo-random numbers $q_{ij} = Enc(dID[s_{ij}], Eid(y, j))$, and send it to the user.

3) Data servers that receive $(ID[y], dID[s_{ij}])$ send the shares W_{ij} corresponding to the ID information back to the user.
4) The user who receives the shares and pseudo-random numbers generated by the servers uses them to restore secret information s_i using the same means as the (k, n) threshold secret sharing method.

ACKNOWLEDGMENT

The authors thank Takato Imai, Honoka Inoue, and Taiki Matsuoka for assistance with the methodology and for sharing their comments and thoughts with them during the course of this research. They thank three reviewers for their insights and comments.

REFERENCES

[1] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
[2] G. R. Blakley, "Safeguarding cryptographic keys," in *Proc. Int. Workshop Manag. Requirements Knowl. (MARK)*, Jun. 1979, pp. 313–318.
[3] J. Kurihara, S. Kiyomoto, K. Fukushima, and T. Tanaka, "A fast (k, L, n)-threshold ramp secret sharing scheme," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 92, no. 8, pp. 1808–1821, 2009.
[4] S. Takahashi, H. Kang, and K. Iwamura, "Asymmetric secret sharing scheme suitable for cloud systems," in *Proc. IEEE 11th Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2014, pp. 784–790.
[5] T. Araki and W. Ogata, "A simple and efficient secret sharing scheme secure against cheating," *IEICE Trans. Fundamentals Electron., Commun. Comput. Sci.*, vol. 94, no. 6, pp. 1338–1345, 2011.
[6] S. Cabello, C. Padró, and G. Sáez, "Secret sharing schemes with detection of cheaters for a general access structure," *Des., Codes Cryptogr.*, vol. 25, no. 2, pp. 175–188, 2002.
[7] W. Ogata and T. Araki, "Cheating detectable secret sharing schemes for random bit strings," *IEICE Trans. Fundamentals Electron., Commun. Comput. Sci.*, vol. 96, no. 11, pp. 2230–2234, 2013.
[8] H. Hoshino and S. Obana, "Almost optimum secret sharing schemes with cheating detection for random bit strings," in *Proc. 10th Int. Workshop Adv. Inf. Comput. Secur. (IWSEC)*, vol. 9241, 2015, pp. 213–222.
[9] L. Zhu, Y. Liu, Y. Wang, W. Ji, X. Hei, Q. Yao, and X. Zhu, "Efficient (k, n) secret sharing scheme secure against k—2 cheaters," in *Proc. Int. Conf. Netw. Netw. Appl. (NaNA)*, Oct. 2017, pp. 1–4.
[10] Q. Al Mahmoud, "A novel verifiable secret sharing with detection and identification of cheaters' group," *Int. J. Math. Sci. Comput.*, vol. 2, no. 2, pp. 1–13, Apr. 2016.
[11] T. P. Pederson, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 576, J. Feigenbaum, Ed. 1992, pp. 129–140.
[12] M. Carpentieri, "A perfect threshold secret sharing scheme to identify cheaters," *Des., Codes Cryptogr.*, vol. 5, no. 3, pp. 183–187, May 1995.
[13] R. J. McEliece and D. V. Sarwate, "On sharing secrets and Reed–Solomon codes," *Commun. ACM*, vol. 24, no. 9, pp. 583–584, Sep. 1981.
[14] W. Ogata, K. Kurosawa, and D. R. Stinson, "Optimum secret sharing scheme secure against cheating," *SIAM J. Discrete Math.*, vol. 20, no. 1, pp. 79–95, Jan. 2006.
[15] M. Carpentieri, A. De Santis, and U. Vaccaro, "Size of shares and probability of cheating in threshold schemes," in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, 1994, pp. 118–125.
[16] L. Harn and C. Lin, "Strong (n,t,n) verifiable secret sharing scheme," *Inf. Sci.*, vol. 180, no. 16, pp. 3059–3064, Aug. 2010.
[17] D. Chen, W. Lu, W. Xing, and N. Wang, "An efficient verifiable threshold multi-secret sharing scheme with different stages," *IEEE Access*, vol. 7, pp. 107104–107110, 2019.
[18] M. H. Dehkordi and S. Mashhadi, "An efficient threshold verifiable multi-secret sharing," *Comput. Standards Interfaces*, vol. 30, no. 3, pp. 187–190, Mar. 2008.
[19] L. Harn and C. Lin, "Detection and identification of cheaters in (t, n) secret sharing scheme," *Des., Codes Cryptogr.*, vol. 52, no. 1, pp. 15–24, Jul. 2009.
[20] T. Araki, "Efficient (k, n) threshold secret sharing schemes secure against cheating from n^{-1} cheaters," in *Proc. 12th Australas. Conf. Inf. Secur. Privacy (ACISSP)*, 2007, pp. 133–142.
[21] B. Rajabi and Z. Eslami, "A verifiable threshold secret sharing scheme based on lattices," *Inf. Sci.*, vol. 501, pp. 655–661, Oct. 2019.
[22] L. J. Pang and Y. M. Wang, "A new (t,n) multi-secret sharing scheme based on Shamir's secret sharing," *Appl. Math. Comput.*, vol. 167, no. 2, pp. 840–848, Aug. 2005.
[23] S. Dutta and R. Safavi-Naini, "Leakage resilient cheating detectable secret sharing schemes," in *Information Security and Privacy* (Lecture Notes in Computer Science), vol. 13083, J. Baek and S. Ruj, Eds. Cham, Switzerland: Springer, 2021, pp. 3–23.
[24] J. Pramanik, S. Dutta, P. S. Roy, and A. Adhikari, "Cheating detectable ramp secret sharing with optimal cheating resiliency," in *Information Systems Security* (Lecture Notes in Computer Science), vol. 12553, S. Kanhere, V. T. Patil, S. Sural, and M. S. Gaur, Eds. Cham, Switzerland: Springer, 2020, pp. 169–184.
[25] C. Yan, Z. Li, L. Liu, and D. Lu, "Cheating identifiable (k, n) threshold quantum secret sharing scheme," *Quantum Inf. Process.*, vol. 21, no. 1, pp. 1–24, Jan. 2022.
[26] S. Kandar and B. C. Dhara, "A verifiable secret sharing scheme with combiner verification and cheater identification," *J. Inf. Secur. Appl.*, vol. 51, Apr. 2020, Art. no. 102430.
[27] S. Gao, "A new algorithm for decoding Reed–Solomon codes," in *Communications, Information and Network Security* (Springer International Series in Engineering and Computer Science), vol. 712, V. K. Bhargava, H. V. Poor, V. Tarokh, and S. Yoon, Eds. Boston, MA, USA: Springer, 2003, pp. 55–68.
[28] J. C. Benaloh, "Secret sharing homomorphisms: Keeping shares of a secret secret (extended abstract)," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 263, A. M. Odlyzko, Ed. Berlin, Germany: Springer, 1987, pp. 251–260.
[29] M. Tompa and H. Woll, "How to share a secret with cheaters," *J. Cryptol.*, vol. 1, no. 3, pp. 133–138, Oct. 1989.

KEIICHI IWAMURA (Member, IEEE) received the B.S. and M.S. degrees in information engineering from Kyushu University, Japan, in 1980 and 1982, respectively, and the Ph.D. degree from The University of Tokyo.

From 1982 to 2006, he was with Canon Inc. He is currently a Professor with the Tokyo University of Science. His research interests include coding theory, information security, and digital watermarking. He is a fellow of the Information Processing Society of Japan. He is the Chairperson of the Technical Committee of Information Hiding and its Criteria for Evaluation and the Technical Committee of Enriched Multimedia, Institute of Electronics, Information and Communication Engineers, Japan.

AHMAD AKMAL AMINUDDIN MOHD KAMAL (Member, IEEE) was born in Penang, Malaysia, in 1994. He received the B.S. and M.S. degrees in electrical engineering and the Ph.D. degree in engineering from the Tokyo University of Science, Japan, in 2017, 2019, and 2022, respectively.

He is currently an Assistant Professor with the Tokyo University of Science. His research interests include information security, multiparty computation using secret sharing, and its application into searchable encryption.