# Secrecy Computation without Changing Polynomial Degree in Shamir's (K, N) Secret Sharing Scheme

Takeshi Shingu[1], Keiichi Iwamura[1] and Kitahiro Kaneda[2]
[1]*Tokyo University of Science, Tokyo, Japan*
[2]*Institute of Document Analysis and Knowledge Science, Osaka Prefecture University, 1-1 Naka-ku, Sakai, Osaka, Japan*

Keywords: Secrecy Computation, Secrecy Multiplication, Secrecy Division, Secrecy Addition, Secrecy Subtraction, (K, N) Secret Sharing Scheme.

Abstract: In This Paper, We Propose a New Secrecy Multiplication Scheme without Changing the Degree in Shamir's (K, N) Secret Sharing Scheme. This Scheme Generates a Scalar Value Called Concealed Secret, Which Multiplies a Secret by a Random Number, and Distributes the Concealed Secret by using a Secret Sharing Scheme. When Secrecy Multiplying, We Temporarily Reconstruct the Concealed Secret, and Multiply It with a Share. Therefore, We Can Perform Secrecy Multiplication without Changing the Degree of Polynomials by Multiplying a Polynomial and Scalar Value. Our Scheme Can Extend to Secrecy Division by Dividing a Share with the Concealed Secret. in Addition, We Propose Secrecy Addition and Subtraction Schemes. We Evaluate the Security of Our Schemes, and Show a Possible Application That Cannot Realized using the Conventional Scheme.

## 1 INTRODUCTION

Cloud computing (Mell, 2011) has brought about considerable changes in users' data utilization. Users can save their data on a server in a network instead of a self-managed server, and can access it from anywhere via a network. However, this incurs some security risks, including server or network failure, in which the users cannot access their data stored on the cloud system. Furthermore, because attacks also are concentrated on the data storage location, the risk of information leakage increases. In particular, in situation where confidential business information is compromised, the leak can cause serious damage.

To counteract those risks, data encryption is recommended. In addition, the saved encryption data is often assumed to be applicable to secrecy calculations without the recovery of the secret data in the cloud system. Therefore, some cloud systems consider applying the "secret sharing scheme" (Shamir, 1979), (Blakley, 1984) to solve the abovementioned problems.

Shamir's $(k, n)$ secret sharing scheme (Shamir, 1979) is a prototypical secret sharing scheme, that distributes n shares of a secret and recovers the secret from $k$ shares. This implies that no secret is revealed if $k$ shares are not revealed, and a secret can be restored even if $n - k$ shares are lost because of a server or network failure. In addition, the secrecy calculation based on the scheme is performed at high speed. Therefore, Shamir's $(k, n)$ secret sharing scheme is suitable for cloud computing systems.

Secrecy calculation (Asharov, 2012), (Beaver, 1991), (Ben-Sasson, 2011), (Ben-Or, 1988) is a technique for performing a computation while keeping the input data secret. It is well known that secrecy addition and subtraction can be easily realized using Shamir's $(k, n)$ secret sharing scheme. However, in secrecy multiplication, the degree of polynomial would change from $k - 1$ to $2k - 2$ because a multiplication of shares is a multiplication between polynomials with a degree of $k - 1$. Therefore, the threshold value $k$ changes only when secrecy multiplication is performed.

In this paper, we propose a new secrecy multiplication scheme without changing the degree of polynomials. The scheme generates a scalar value called concealed secret, which multiplies a secret by a random number, and distributes the concealed secret by using a secret sharing scheme. When multiplying, we temporarily reconstruct the concealed secret and multiply it with a share. Thus, we can perform secrecy multiplication without



Shingu, T., Iwaumura, K. and Kaneda, K.
Secrecy Computation without Changing Polynomial Degree in Shamir's (K, N) Secret Sharing Scheme.
DOI: 10.5220/0005998800890094
In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 1: DCNET*, pages 89-94
ISBN: 978-989-758-196-0

changing the degree of a polynomial, because the multiplication of two polynomials is deformed during the multiplication of a polynomial and a scalar value. Our scheme can be extended secrecy division by dividing a share by the concealed secret. In addition, we propose the secrecy addition and subtraction schemes by considering the concealed secret, and evaluate the security of our schemes, and show the possible application that cannot be realized using our schemes.

The remainder of this paper is organized as follows. In section 2I, we describe the conventional secrecy computation. In section 3, we propose new secrecy multiplication, division, addition and subtraction schemes. In section 4, we evaluate the security of our schemes and discuss the possible application. Finally, we summarize the results in section 5.

# 2 CONVENTIONAL SCHEME

## 2.1 Shamir's (K, N) Secret Sharing Scheme

In this paper, $s$ is a secret to be distributed, $n$ is the number of players, and $k$ is the threshold to restore the secret.

[Distribution]

1. The dealer selects a prime number $\boldsymbol{p}$ that satisfies $\boldsymbol{s} < \boldsymbol{p}$ and $\boldsymbol{n} < \boldsymbol{p}$.
2. The dealer chooses $\boldsymbol{n}$ elements of GF($\boldsymbol{p}$) and assigns them to each player as $\boldsymbol{x_i}(\boldsymbol{i} = \boldsymbol{1, 2, \cdots, n})$.
3. The dealer generates equation $\boldsymbol{W(x)}$ from secret $\boldsymbol{s}$ and $\boldsymbol{k-1}$ random numbers, that is, $\boldsymbol{a_l}(\boldsymbol{l} = \boldsymbol{1, 2, \cdots, k-1})$ selected from elements of GF($\boldsymbol{p}$).

$W(x) = s + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1} \pmod p$

4. The dealer calculates $\boldsymbol{W_i} = \boldsymbol{W(x_i)}$ $(\boldsymbol{i} = \boldsymbol{1, 2, \cdots, n})$ and distributes them to each player $\boldsymbol{P_i}$.

[Reconstruction]

1. The shares that are used for reconstruction are assumed to be $\boldsymbol{W_i}(\boldsymbol{i} = \boldsymbol{1, 2, \cdots, k})$.

The user who reconstructs the secret s obtains s by solving $k$ simultaneous equations by using the $k$ shares.

## 2.2 Secrecy Computation

Let $a$ $and$ $b$ be two secrets that are shared using polynomials $f(x)$ $and$ $g(x)$, which are of degree $k-1$. Each player $P_i$ has $f(x_i)$ $and$ $g(x_i)$. Let $c$ ($\neq 0$) be a scalar value. The secrecy addition $a+b$ and the secrecy multiplication $c \times a$ are easy to perform because the degree of the polynomials that is obtained as a result remains unchanged.

[Secrecy addition and subtraction]

1. $P_i$ computes $h(x_i) = f(x_i) + g(x_i)$, which is the share of $a+b$, and sends it to the user.
2. The user who reconstructs the result of $a+b$ solves $k$ simultaneous equations by using $k$ shares.

Similarly, secrecy subtraction can be realized by changing "+" to "−".

[Secrecy multiplication]

Secrecy multiplication using $f(x_i)$ $and$ $g(x_i)$ is not as simple as addition because the multiplication of $f(x_i)$ and $g(x_i)$ of degree $k-1$ will result in $h(x)$ of degree $2k-2$. Therefore, a degree reduction and a randomization of $h(x)$ is required.

*The reduction step*

Let $h(x) = h + h_0 x + \cdots + h_{2k-2} x^{2k-2}$ and $s_i = h(x_i) = f(x_i) g(x_i)$. Each $P_i$ has $s_i$. Define the truncation of $h(x)$ to be $k(x) = h + h_0 x + \cdots + h_{k-1} x^{k-1}$, and $r_i = k(x_i)$.

Let $S = (s_0, \cdots, s_{n-1})$ and $R = (r_0, \cdots, r_{n-1})$, then there is a constant $n \times n$ matrix A such that $R = S \cdot A$.

Let H be an n-vector such that

$$H = (h_0, \cdots, h_{k-1}, \cdots, h_{2k-2}, 0, \cdots, 0)$$

and let K be an n-vector such that

$$K = (h_0, \cdots, h_{k-1}, 0, \cdots, 0).$$

Let $B = (b_{i,j})$ be the $n \times n$ (Vandermonde) matrix, where $b_{i,j} = x_j^i$ for $i, j = 0, \cdots, n-1$. Furthermore let the linear projection $P = (x_0, \cdots, x_{n-1}) = (x_0, \cdots, x_{k-1}, 0, \cdots, 0)$. Then, we have

$$H \cdot B = S$$
$$H \cdot P = K$$
$$K \cdot B = R$$
$$S \cdot (B^{-1} P B) = R$$

*The randomization step*

To randomize the coefficients of the polynomial, each $P_i$ randomly selects a polynomial $q_i(x)$ of degree $2k-2$ with a zero free coefficient and distributes its shares among the players. Thus, instead of using $h(x)$ in this reduction players can use

$$\bar{h}(x) = h(x) + \sum_{j=0}^{n-1} q_j(x)$$

Which satisfies $\bar{h}(0) = h(0)$ ; however, the other coefficients of $x^i, 1 \le i \le k-1$ are completely random.

# 3 PROPOSED SCHEME

Our scheme can be applied to any homomorphic $(k, n)$ secret sharing scheme. Therefore, we used Shamir's secret sharing scheme. In this section, we describe the distribution, reconstruction, and secrecy multiplication, division, addition, and subtraction. The variables $a$ and $b$ are secrets and are elements of GF( $p$ ). $\alpha_j$ and $\beta_j$ are random numbers and are selected from the elements of GF($q$). The variables $p$ and $q$ are primes, and $q \geqq 2p$ . The secrets and random numbers are non-zero. Computations are performed on the field of GF(q). Our scheme assumes a semi-honest model, and all players follow our scheme.

## 3.1 Notation

- $\overline{[a]}_i$: A share of $a$ for player $P_i$ by using secret sharing scheme
- $[a]_i$ : A set of shares, such as $\left(\overline{[\alpha a]}_i, \overline{[\alpha_0]}_i, \cdots, \overline{[\alpha_{k-1}]}_i\right)$, on $a$ for player $P_i$.
- $SHR(\alpha a) = \left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{n-1}\right)$ : Distributing $\alpha a$ to the shares $\left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{n-1}\right)$.
- $REC\left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{k-1}\right) = \alpha a$ : Reconstructing $\alpha a$ from the shares $\left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{n-1}\right)$.

## 3.2 Distribution

Input : $a$
Output : $[a]_i \coloneqq \left(\overline{[\alpha a]}_i, \overline{[\alpha_0]}_i, \cdots, \overline{[\alpha_{k-1}]}_i\right)$ $(i = 0,1, \cdots, n-1)$

1. The dealer picks $k$ random numbers $\alpha_0, \cdots, \alpha_{k-1}$ and computes $\alpha$ as follows:

$$\alpha = \prod_{j=0}^{k-1} \alpha_j$$

2. The dealer then computes $\alpha a$ as a concealed secret and distributes $(\alpha a, \alpha_0, \cdots, \alpha_{k-1})$ to $n$ players by using the $(k, n)$ secret sharing scheme.

$$SHR(\alpha a) = \left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{k-1}\right)$$
$$SHR(\alpha_0) = \left(\overline{[\alpha_0]}_0, \cdots, \overline{[\alpha_0]}_{k-1}\right)$$
$$\vdots$$
$$SHR(\alpha_{k-1}) = \left(\overline{[\alpha_{k-1}]}_0, \cdots, \overline{[\alpha_{k-1}]}_{k-1}\right)$$

3. Player $P_i$ has $[a]_i \coloneqq \left(\overline{[\alpha a]}_i, \overline{[\alpha_0]}_i, \cdots, \overline{[\alpha_{k-1}]}_i\right)$ as shares on secret $a$.

## 3.3 Reconstruction

Input : $[a]_j \coloneqq \left(\overline{[\alpha a]}_j, \overline{[\alpha_0]}_j, \cdots, \overline{[\alpha_{k-1}]}_j\right)$ $(j = 0,1, \cdots, k-1)$
Output : $a$

1. A user who restores secret $a$ collects $[a]_j$ from $k$ players.
2. The user obtains secret $a$ by reconstructing $\alpha a, \alpha_0, \cdots, \alpha_{k-1}$ as follows,

$$REC\left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{k-1}\right) = \alpha a$$
$$REC\left(\overline{[\alpha_0]}_0, \cdots, \overline{[\alpha_0]}_{k-1}\right) = \alpha_0$$
$$\vdots$$
$$REC\left(\overline{[\alpha_{k-1}]}_0, \cdots, \overline{[\alpha_{k-1}]}_{k-1}\right) = \alpha_{k-1}$$
$$\alpha = \prod_{j=0}^{k-1} \alpha_j$$
$$\alpha a \times \alpha^{-1} = a$$

## 3.4 Secrecy Multiplication

We assume that player $P_j$ $(j = 0,1, \cdots, n-1)$ has $[a]_j$ and $[b]_j$ for secrets $a$ and $b$, respectively.

From $[c]_i (i = 0,1, \cdots, k-1)$ of the output, we can restore $c = ab$ through reconstruction shown in section 3.3.

Input : $[a]_j, [b]_j$ $(j = 0,1, \cdots, k-1)$
Output : $[c]_i \coloneqq [ab]_i$ $(i = 0,1, \cdots, n-1)$

1. $P_0$ collects $\overline{[\alpha a]}_j$ from $k$ players.
2. $P_0$ reconstructs $\alpha a$ and sends it to all players.

$$REC\left(\overline{[\alpha a]}_0, \cdots, \overline{[\alpha a]}_{k-1}\right) = \alpha a$$

3. $P_j$ computes $\overline{[\alpha a \beta b]}_j$ by multiplying the share $\overline{[\beta b]}_j$ by $\alpha a$.
4. $P_j$ collects $\overline{[\alpha_j]}_0, \cdots, \overline{[\alpha_j]}_{k-1}$ and $\overline{[\beta_j]}_0, \cdots, \overline{[\beta_j]}_{k-1}$ from $k$ players and reconstructs $\alpha_j$ and $\beta_j$.
5. $P_j$ computes $\alpha_j \beta_j$ and performs $SHR(\alpha_j \beta_j)$.
6. $P_i$ has $[c]_i \coloneqq \left(\overline{[\alpha\beta ab]}_i, \overline{[\alpha_0\beta_0]}_i, \cdots, \overline{[\alpha_{k-1}\beta_{k-1}]}_i\right)$ , where $\overline{[\alpha a \beta b]}_j$ is a randomized share.

## 3.5 Secrecy Division

Input : $[a]_j, [b]_j$ $(j = 0,1, \cdots, k-1)$

Output : $[c]_i \coloneqq [b/a]_i$ $(i = 0,1,\cdots,n-1)$

1. $P_0$ collects $\overline{[\alpha a]}_j$ from $k$ players.
2. $P_0$ reconstructs $\alpha a$ and sends it to all players.
3. $P_j$ computes $\overline{[\beta b/\alpha a]}_j$ by dividing the share $\overline{[\beta b]}_j$ by $\alpha a$.
4. $P_j$ collects $\overline{[\alpha_j]}_0,\cdots,\overline{[\alpha_j]}_{k-1}$ and $\overline{[\beta_j]}_0,\cdots,\overline{[\beta_j]}_{k-1}$ from $k$ players and reconstructs $\alpha_j$ and $\beta_j$.
5. $P_j$ computes $\beta_j/\alpha_j$ and performs $SHR(\beta_j/\alpha_j)$.
6. $P_i$ has $[c]_i \coloneqq \left(\overline{[\beta b/\alpha a]}_i, \overline{[\beta_0/\alpha_0]}_i,\cdots,\overline{[\beta_{k-1}/\alpha_{k-1}]}_i\right)$ where $\overline{[\beta b/\alpha a]}_j$ is a randomized share.

### 3.6 Secrecy Addition and Subtraction

It is possible to perform secrecy addition and subtraction by using the concealed secret, although it is not efficient compared to the conventional scheme. The computation of $[a+b]_i$ from $[a]_j$ and $[b]_j$ for secrets $a$ and $b$ is as follows.

Input: $[a]_j, [b]_j$ $(j = 0,1,\cdots,k-1)$

Output: $[c]_i \coloneqq [a+b]_i$ $(i = 0,1,\cdots,n-1)$

1. $P_j$ collects $\overline{[\alpha_j]}_0,\cdots,\overline{[\alpha_j]}_{k-1}$ and $\overline{[\beta_j]}_0,\cdots,\overline{[\beta_j]}_{k-1}$, and reconstructs $\alpha_j, \beta_j$. $P_j$ picks a random number $\gamma_j \in \boldsymbol{Z}/p\boldsymbol{Z}$ and sends $\alpha_j\,\gamma_j$ and $\beta_j\gamma_j$ to $P_0$.
2. $P_0$ reconstructs $\alpha\gamma$ and $\beta\gamma$, and sends them to all players.

$$\alpha\gamma = \prod_{j=0}^{k-1} \alpha_j\gamma_j$$

$$\beta\gamma = \prod_{j=0}^{k-1} \beta_j\gamma_j$$

3. $P_j$ computes the share of $\alpha\beta\gamma(a+b)$ as follows:

$$\overline{[\alpha\beta\gamma(a+b)]}_j = \alpha\gamma \cdot \overline{[\beta b]}_j + \beta\gamma \cdot \overline{[\alpha a]}_j$$

4. $P_j$ computes $\alpha_j\beta_j\gamma_j$ and performs $SHR(\alpha_j\beta_j\gamma_j)$.
5. $P_i$ holds

$[c]_i \coloneqq \left(\overline{[\alpha\beta\gamma(a+b)]}_i, \overline{[\alpha_0\beta_0\gamma_0]}_i,\cdots,\overline{[\alpha_{k-1}\beta_{k-1}\gamma_{k-1}]}_i\right)$

Similarly, the secrecy subtraction can be realized by changing "+" to "−".

## 4 SECURITY EVALUATION AND DISCUSSION

### 4.1 Security Analysis of Secrecy Multiplication and Division

In secrecy multiplication, all players know $\alpha a$, but cannot know $a$, because $\alpha$ is a random number. Namely, the expression below is realized, where $H(x)$ shows the entropy of $x$.

$$H(a) = H(a|\alpha a)$$

Even if $P_j$ $(j = 0,1,\cdots,k-2)$ colludes, $\alpha$ and $\beta$ are not revealed, although $P_j$ knows $\alpha_j$ and $\beta_j$.

$$H(\alpha) = H(\alpha|\alpha_0,\ldots,\alpha_{k-2})$$
$$H(\beta) = H(\beta|\beta_0,\ldots,\beta_{k-2})$$

Next, because $\alpha a$ is multiplied by the share $\overline{[\beta b]}_j$, $\alpha a\beta b$ is not revealed even if $k-1$ players collude.

$$H(\alpha a\beta b) = H(\alpha a\beta b|\alpha a, \overline{[\beta b]}_0,\ldots,\overline{[\beta b]}_{k-2})$$

In addition, $ab$ is not revealed, because $a\beta$ is not known.

$$H(a\beta) = H(a\beta|\alpha_0\beta_0,\ldots,\alpha_{k-2}\beta_{k-2})$$

Let $A$ denote an arbitrary set of participants such that $|A| \leq k-1$. Then, we have

$$H(\alpha) = H(\alpha|VA)$$
$$H(\beta) = H(\beta|VB)$$
$$H(a\beta) = H(a\beta|VAB)$$

where $VA$ denotes a set of shares on $a$ that are given to each participant in $A$, $VB$ denotes a set of shares on $b$ given to each participant in $A$, and $VAB$ denotes a set of shares on $a$ and $b$ given to each participant in $A$.

Alternatively, let $A$ denote an arbitrary set of participants such that $|A| \geq k$. Then, it is clear that the reconstruction algorithm can recover the secret from the shares given to each participant in $A$.

In the reconstruction, although the user who restores the result $ab$ knows $a\beta$ and $a\beta ab$, he cannot know $a$ or $b$. Even if the user and $k-1$ players collude, they cannot know $a$ or $b$.

Secrecy division has the same security.

### 4.2 Security Analysis of Secrecy Addition and Subtraction

In secrecy addition, all players know $\alpha\gamma$ and $\beta\gamma$. In the reconstruction, the user who restores $a+b$ knows $\alpha\beta\gamma$ and $\alpha\beta\gamma(a+b)$. Therefore, if a user and

a player collude, $\alpha$ and $\beta$ are known. However, $\alpha a$ or $\beta b$ cannot be reconstructed, even if the user and $k-1$ players collude. Therefore, they cannot know $a$ or $b$.

Secrecy subtraction has the same security.

## 4.3 Combination of Secrecy Addition and Multiplication

In our scheme, a simple combination of secrecy multiplication and addition features a problem.

Input: $[a]_j, [b]_j$ $(j = 0,1,\cdots,k-1)$

Output: $[c]_i \coloneqq [ab]_i,\ [d]_i \coloneqq [a+b]_i (i = 0,1,\cdots,n-1)$

1. All players perform secrecy multiplication, as shown in section 3.4, and obtain $[c]_i$.
2. All players perform secrecy addition, as shown in section 3.6, and obtain $[d]_i$.

Therefore, $\alpha a$ is known in secrecy multiplication, and $\alpha$ and $\beta$ are known in secrecy addition. Therefore, $a$ and $b$ are known.

To solve this problem, we prepare one or more sets for a secret. $[a]_j^l$ represents a different set of shares for secret $a$ by using different random numbers $\alpha_j^l$ selected independently (as in section 3.2). For example, when $\alpha_j^1$ and $\alpha_j^2$ $(i = 0,\cdots,k-1)$ are uniform random numbers, $[a]_i^1$ and $[a]_i^2$ are expressed as follows;

$$[a]_i^1 = \left(\overline{[\alpha^1 a]_i}, \overline{[\alpha_0^1]_i}, \cdots, \overline{[\alpha_{k-1}^1]_i}\right)$$

$$[a]_i^2 = \left(\overline{[\alpha^2 a]_i}, \overline{[\alpha_0^2]_i}, \cdots, \overline{[\alpha_{k-1}^2]_i}\right)$$

$$\left(\alpha^1 = \prod_{j=0}^{k-1} \alpha_j^1, \alpha^2 = \prod_{j=0}^{k-1} \alpha_j^2\right)$$

By using $[a]_j^l$, we can combine secrecy multiplication and addition as follows, (where $[a]_j^l$ is deleted after its used).

Input: $[a]_j^1, [a]_j^2, [b]_j^1, [b]_j^2$ $(j = 0,1,\cdots,k-1)$

Output: $[c]_i \coloneqq [ab]_i,\ [d]_i \coloneqq [a+b]_i (i = 0,1,\cdots,n-1)$

1. All players perform secrecy multiplication, as shown in section 3.4, by using $[a]_j^1$ and $[b]_j^1$, and obtain $[c]_i$.
2. All players perform secrecy addition as shown in section 3.6, by using $[a]_j^2$ and $[b]_j^2$, and obtain $[d]_i$.
3. All players delete $[a]_j^2$, $[b]_j^2$, $[a]_j^2$, and $[b]_j^2$.

In this case, because the $\alpha$ of $\alpha a$ in secrecy multiplication and the $\alpha$ in secrecy addition are different, $a$ is not revealed.

In the case where $[z]_j^1$ calculated using $[a]_j^1$,…, $[y]_j^1$ is used twice, such as in a square calculation, $[z]_j^2$ calculated from $[a]_j^2$,…, $[y]_j^2$ is used as $[z]_j^1 \times [z]_j^2$.

Therefore, secrecy multiplication, division, addition, subtraction, and the combination of secrecy multiplication (division) and secrecy addition (subtraction) have information theoretical security.

## 4.4 Possible Application

As the conventional scheme requires $2k-1 \leqq n$, the value of $n$ must be approximately twice that of $k$. In a cloud system, because $n$ is the number of servers that store the shares, the composition of the cloud system is restricted. For example, $(k,n) = (2,2)(3,3)(3,4)\cdots$cannot be selected. In contrast, our scheme can select $k$ and $n(\leqq k)$ without restriction.

As one application, we consider the diagnostic data for patients such as their blood glucose level and the value of hemoglobin. Such values do not take 0. We consider the case where a medical company wants to maintain the average blood glucose level of certain number of patients in different hospitals, keeping the sum of blood glucose level and total number of patients secret. Let the number of hospitals be three. Further, let $a1$, $b1$, and $c1$ be the blood glucose levels, and $a2$, $b2$, and $c2$ be the number of patients in hospitals A, B, and C, respectively. In this case, the (3,3) secret sharing scheme is suitable. When the conventional scheme is used, hospital A distributes $\overline{[a1]}_j$ and $\overline{[a2]}_j$, hospital B distributes $\overline{[b1]}_j$ and $\overline{[b2]}_j$, and hospital C distributes $\overline{[c1]}_j$and $\overline{[c2]}_j$ to the other two hospitals, where j = A, B, C. Each hospital calculates $\overline{[a1+b1+c1]}_j$ and $\overline{[a2+b2+c2]}_j$, respectively. To obtain the average, it is necessary to divide $\overline{[a1+b1+c1]}_j$ by $\overline{[a2+b2+c2]}_j$. However, division of shares is difficult through the conventional scheme. If the three hospitals send the shares to the company, and the company can restore $a1+b1+c1$ and $a2+b2+c2$, it can obtain the average by dividing them. However, the company knows the sum of blood glucose level and the total number of patients of the three hospitals. Even if the shares of the multiplicative inverse of the total number are obtained, the conventional scheme cannot calculate the multiplication of shares because of (k,n)=(3,3).

In contrast, our schemes can perform secrecy addition and division by using (3,3) secret sharing scheme. In this case, each hospital has $[a1]_j^1$, $[b1]_j^1$,

$[c1]_j^1$, $[a2]_j^1$, $[b2]_j^1$, and $[c2]_j^1$ (j = A, B, C), and calculates $[a1 + b1 + c1]_j^1$ and $[a2 + b2 + c2]_j^1$ by using secrecy addition as shown in section 3.6. The average is obtained using secrecy division as shown in section 3.5, or using secrecy multiplication with the shares of the multiplicative inverse as shown in section 3.4. If other secrecy calculations are needed, $[a1]_j^1$, $[b1]_j^1$, $[c1]_j^1$, $[a2]_j^1$, $[b2]_j^1$, and $[c2]_j^1$ are never used, instead $[a1]_j^2$, $[b1]_j^2$, $[c1]_j^2$, $[a2]_j^2$, $[b2]_j^2$, and $[c2]_j^2$ are generated and used.

## 5 CONCLUSIONS

We proposed a new secrecy multiplication scheme without changing the degree of the polynomials in a $(k, n)$ secret sharing scheme. In this scheme, we can set $k = n$ in secrecy multiplication. This scheme has information theoretical security, and can be extended to secrecy division, addition, and subtraction. Our new schemes realize some applications that were not possible by using the conventional scheme.

## REFERENCES


Shamir, A. 1979. *How to share a secret.* Communications of the ACM, 22, (11), pp. 612-613.

Blakley, G. R. 1984. Security of ramp schemes. CRYPTO ’84, pp. 242-268.

Mell, P., Grance, T. 2011. *The NIST Definition of Cloud Computing.* National Institute of Standards and Technology.

Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D. 2012. *Secrecy computation with low communication, computation and interaction via threshold FHE.* In D. Pointcheval and T. Johansson, editors, EUROCRYPT, volume 7237 of Lecture Notes in Computer Science, pp. 483–501. Springer.

Beaver, D., 1991. *Efficient secrecy protocols using circuit randomization.* In J. Feigenbaum, editor, CRYPTO, volume 576 of Lecture Notes in Computer Science, pp. 420–432. Springer.

Ben-Sasson, E., Fehr, S., Ostrovsky, R. 2011. *Near-linear unconditionally-secure secrecy computation with a dishonest minority.* IACR Cryptology ePrint Archive, 2011:629.

Ben-Or, M., Goldwasser, S., Wigderson, A. 1988. *Completeness theorems for non-cryptographic fault-tolerant distributed computation.* Communications of the ACM, pp. 1-10.

Krawczyk, H. 1994. *Secret sharing made short.* CRYPTO ’93, pp. 136-146.

Kawamoto, Y., Yamamoto, H. 1985. *(k,L,n) Ramp secret sharing systems for functions.* IEIC, vol. J68-A, no. 9, pp. 945-952.

Ito, M., Saito, A., Nishizeki, T. 1987. *Secret sharing scheme realizing general access structure.* Proceedings of the IEEE Global Telecommunications Conference, Globecom ’87, pp. 99-102.