

非対称秘密分散法に適した分散値の更新手法

金子 直人^{†1} 岩村 恵市^{†1}

概要：クラウドコンピューティングに (k,n) 閾値秘密分散法を適用することが考えられている。しかし、Shamir 法などの従来の秘密分散法は、 k 個の分散値が漏洩すれば秘密情報は漏洩するが、 k 個の分散値を漏洩させない仕組みを持たない。それに対して、秘密情報のオーナーが一つの鍵から $k-1$ までの分散値を生成できる非対称秘密分散法は、外部サーバの数を k 未満とすることができるため、攻撃者が全ての外部サーバを攻撃しても秘密情報が漏洩しない仕組みを持つ。また、非対称秘密分散法はオーナーが分散値を生成しなければ秘密情報を復元できないため、オーナーによる秘密情報の復元制御が可能である。しかし、オーナーが管理する鍵が盗難・漏洩した場合、秘密情報が不正に復元される可能性がある。そこで本論文では、上記非対称秘密分散法に適した分散値の更新法を提案する。提案法は、全ての外部サーバが解析されても、更新情報を含め秘密情報が漏洩しない。さらに、従来の分散値更新法に比べて、更新に必要な通信量を大幅に削減する。さらに、従来の分散値更新法の問題である $k-1$ 個のサーバ解析によって更新情報が漏洩するという問題点を解決する方法も示す。

キーワード：プロアクティブ秘密分散法，非対称秘密分散法

Proactive secret sharing scheme suitable for asymmetric secret sharing scheme

NAOTO KANEKO^{†1} KEIICHI IWAMURA^{†1}

Abstract:

Keywords: Proactive Secret Sharing, Asymmetric Secret Sharing

1. はじめに

近年、クラウドコンピューティング[1]の利用やアプリの開発が盛んに行われている。クラウドは従来ユーザが自前の装置などに管理していたデータを、オンライン上のサーバに保有、管理することによって、ユーザは自身のデータを持ち歩くことなしに、ネットワークを介していつでもどこでも必要なデータにアクセスできるようになる。また、クラウド上に大量のデータを保存しておくこともできる。さらに、短期間で導入が可能というメリットもある。しかしながら、クラウドを使用する際には、基本的にはすべてのデータがクラウドに集約されるため、安全性に関して気を付けなければならない問題点がいくつかある。その一つは、サーバやネットワークの障害などによって、クラウドサービスが利用できなくなってしまう場合である。さらに集中的なデータの管理は攻撃の標的となりやすく、情報流出の危険性が高まる。特に、企業の機密情報などが流出した場合には、致命的な被害を生じさせる場合もある。また、情報を守るために通常暗号化を行うが、秘匿計算まで考慮に入れた場合、一般に準同型性を持つ暗号化は処理が重く、秘匿計算しようとした際に時間がかかってしまう。以上の問題を解決するため、クラウドシステムに「秘密分

散法」を適用することが考えられている。

秘密分散法 (secret sharing scheme) [2] は、一つの情報を n 個の分散値に変換し、そのうちの k 個以上が集まれば元の情報が復元可能だが、 k 個未満では元の情報は全く復元されないという手法である。これによって、サーバやネットワークの障害などによりデータの一部が使えなくても、それが $n-k$ 個以下ならば元の情報が復元でき、さらに k 個以上の情報が漏洩しない限り情報漏洩は起こらないという安全な情報管理システムが実現できる。

一方、秘密分散法は一つの情報が複数の情報に変換されるため、管理すべき情報の量が増加するという問題点がある。また、一般的なクラウド利用は利用者が 1 つのクラウド業者と契約して情報保管してもらうという場合がほとんどである。よって、そのクラウド業者が秘密分散法を適用していても、そのクラウド業者は全てのサーバにアクセス可能なため秘密情報の復元が可能である。従って、利用者は秘密情報の意図しない復元の危惧から逃れられない。

その一つの解決策として高橋らによって提案された非対称秘密分散法がある。非対称秘密分散法とは、Ramp 型秘密分散法のように全サーバの記憶容量を均等に削減するのではなく、特定のサーバの持つ分散値を削減して非均一に記憶容量を削減する手法である。分散値を削減されたサーバ

^{†1} 東京理科大学
Tokyo University of Science

は鍵サーバと呼ばれ、管理する情報を鍵のみにすることができる。また、非対称秘密分散法も準同性をもつため、秘密情報を秘匿したまま演算を行う秘匿演算を実行できる。ここで、秘密情報のオーナー自身が鍵サーバとなり鍵を自ら管理すればオーナー自身が秘密情報の復元や秘匿演算への利用などを制御することが可能である。この場合、管理すべき秘密情報が膨大でもオーナーは鍵のみを管理すればよいためスマホなどの携帯端末を鍵サーバとすることができる。

このような応用においては、オーナーが携帯端末をなくすまたは盗難される、壊れるなどの問題が考えられる。オーナーの携帯端末は鍵を管理しているだけなので、鍵のバックアップ値を別の安全なところに保管しておけば新たな携帯端末にそのバックアップ値を再設定することによって秘密情報の分散・復元または秘匿演算を継続して行うことができる。しかし、盗まれるまたは失われた携帯端末はパスワードや生体認証などを用いて厳重に管理されていたとしても、他人の手に渡った可能性がある場合、その携帯端末で管理されていた分散値は全て更新することが望まれる。よって、本論文ではまず非対称秘密分散において効率的に分散値の更新を行う手法を提案する。

一方、非対称秘密分散法に限らず一般の秘密分散法における分散値の更新方法として、情報を復元せずに更新する手法いくつか提案されている。このような手法の一つとして、Shamir の秘密分散法（以降、Shamir 法）を対象にして Herzberg らによる更新手法がよく知られている。この手法は、定数項が 0 となる $k-1$ 次多項式をすべてのサーバが生成し、それを互いに通信し、自身の分散値に加えることで分散値の更新を行う。しかし、 n 台のサーバを想定した場合、 $n \times (n-1)$ の通信が発生し、さらにこれを分散値毎に行うため、1 つのサーバが管理する分散値の数を m とした場合、 $n \times (n-1) \times m$ の通信となる。特に、システムとしての実運用においては通信量がボトルネックとなる場合が多い。

また、Herzberg の更新法では定数項を 0 とする $k-1$ 次多項式を用いるため、 $k-1$ 個の更新値からその多項式が解かれる。よって、攻撃者が $k-1$ 台のサーバを監視できれば更新に用いた多項式が解かれ、全ての更新値が漏洩するという問題も持つ。すなわち、 k を閾値とする秘密分散法において、更新に関する閾値は $k-1$ になるという問題が新たに発生する。そこで本論文では、2 つ目の提案として非対称秘密分散法だけに限らず、更新における閾値も k となる新しい分散値の更新法を示す。

本論文の構成は、第 2 章において Shamir 法及び非対称秘密分散法、Herzberg らによる更新手法などの関連技術について述べ、第 3 章において非対称秘密分散法に適した分散値の更新手法を示し評価する。さらに、第 4 章において非対称秘密分散法に限らず更新の閾値が k となる手法を提案する。

2. 従来方式

2.1 Shamir の (k, n) しきい値秘密分散法

次の 2 つの条件を満たす秘密分散法を (k, n) しきい値秘密分散法と呼ぶ。

1. 秘密情報 s から生成された n 個の分散値のうち、任意の k 個から元の s を復元することができる。
2. $k-1$ 個以下の分散値からは、秘密情報 s に関する情報は一切得ることができない。

Shamir の提案した多項式補間による方法では、以下のようにして (k, n) しきい値秘密分散法を実現する。

[分散]

1. $s < p$ かつ $n < p$ である任意の素数 p を選ぶ。
2. $\mathbf{Z}/p\mathbf{Z}$ から、異なる n 個の $x_i (i = 1, 2, \dots, n)$ を選び出し、サーバ ID とする。
3. $\mathbf{Z}/p\mathbf{Z}$ から、 $k-1$ 個の乱数 $a_l (l = 1, 2, \dots, k-1)$ を選んで、以下の式（以降、分散式と呼ぶ）を生成する。

$$W_i = s + a_1 x_i + a_2 x_i^2 + \dots + a_{k-1} x_i^{k-1} \pmod{p} \quad (1)$$

4. 上記式(1)の x_i に各サーバ ID を代入して、分散値 W_i を計算し、各サーバにこれらのサーバ ID x_i と分散値 W_i を配布する

[復号]

1. 復号に用いる分散値を $W_i (i = 1, 2, \dots, k)$ とする。また、その分散値に対応するサーバ ID を x_i とする。

分散式に x_i と W_i を代入し、 k 個の連立方程式を解いて、 s を得る。 s の復元の際には、Lagrange の補間公式を用いると便利である。

2.2 非対称秘密分散法

クラウドシステムを構成する n 台のサーバから l 台 ($l < k$) を鍵サーバとし、鍵サーバは分散情報を持たないものとする。鍵サーバ以外の $n-l$ 台のサーバをデータサーバと呼び、これらは分散値を生成するオーナーから送られた分散値を保管する。また、秘密情報 $s_i (i = 1, \dots, m)$ のオーナーは自身の鍵 key_0 を持つ。更に、オーナーが持つ m 個の秘密情報 $s_i (i = 1, \dots, m)$ にはそれぞれデータ識別のため $dID[s_i] (i = 1, \dots, m)$ が割り振られているものとする。ただし、条件付きエントロピーの式は以下になるものとする。

$$H(s_i | dID[s_i]) = H(s_i) \quad (2)$$

また、 $Enc(a, b)$ は a を b という鍵を用いて暗号化する処理を表すものとする。

[分散]

1. オーナは自身の鍵 key_0 から以下の式より、鍵サーバに x_j に対する l 個の鍵 $key_j (j = 1, \dots, l)$ を生成する。

$$key_j = Enc(j, key_0) \quad (3)$$

2. オーナは自身の秘密情報に関するデータ識別子 $dID[s_i] (i = 1, \dots, m)$ を用いて式(4)より、擬似乱数 q_{ij} を

生成する.

$$q_{ij} = \text{Enc}(dID[s_i], \text{key}_j) \quad (4)$$

3. オーナはまず式(1)の $k-1$ 次の分散式の係数ベクトル $A(i) = [s_i, a_{i1}, \dots, a_{ik-1}]^T$ における $k-1-l$ 次の部分ベクトル $A'_{k-1-l} = [a_{il+1}, \dots, a_{ik-1}]^T$ を真性乱数を用いて i ごとに定める. その後, 手順(3)で生成した l 次の擬似乱数系列 $Q = [q_{i1}, \dots, q_{il}]^T$ 及び鍵サーバの ID 系列

$$X' = \begin{bmatrix} x_1 & \dots & x_1^{k-1} \\ \vdots & \ddots & \vdots \\ x_l & \dots & x_l^{k-1} \end{bmatrix} \quad (5)$$

を用いて以下の式から分散式の $k-1$ 次の部分ベクトル $A(i)_{k-1} = [a_{i1}, \dots, a_{ik-1}]^T$ における残りの部分ベクトル $A'_l(i) = [a_{il}, \dots, a_{ik-1}]^T$ を i ごとに算出する.

$$\begin{bmatrix} q_{i1} \\ \vdots \\ q_{il} \end{bmatrix} = \begin{bmatrix} s_i \\ \vdots \\ s_i \end{bmatrix} + \begin{bmatrix} x_1 & \dots & x_1^{k-1} \\ \vdots & \ddots & \vdots \\ x_l & \dots & x_l^{k-1} \end{bmatrix} \begin{bmatrix} a_{i1} \\ \vdots \\ a_{ik-1} \end{bmatrix} \quad (6)$$

ここで上記式において $S = [s_i, \dots, s_i]^T$ をとすると,

$$A(i)_{k-1} = X'^{-1}(Q - S) \quad (7)$$

これにより, オーナは $k-1$ 次の分散式の係数ベクトル $A(i) = [s_i, a_{i1}, \dots, a_{ik-1}]^T$ における $k-1$ 次の部分ベクトル $A(i)_{k-1} = [a_{i1}, \dots, a_{ik-1}]^T$ をすべて決定することができる.

4. また, オーナはデータサーバ $x_{l+1}, \dots, x_n$ に関する分散値 $W_{il+1}, \dots, W_{in}$ を手順(4)で生成した係数行列を利用して (k, n) 閾値秘密分散法と同様の手順により算出する.
5. オーナはそれぞれのデータサーバに生成した分散値 $W_{1j}, \dots, W_{mj} (j = l+1, \dots, n)$ を送る.

[復元]

1. 秘密情報 s_i を復元するオーナは n 個のサーバ $x_1, \dots, x_n$ から任意の k 個のサーバを選択し, 選択したサーバに対してデータ識別子 $dID[s_i]$ を送る.
2. 鍵サーバ x_j が選択された場合, オーナは式(4)より擬似乱数 q_{ij} を生成する.
3. データサーバが選択された場合, $dID[s_i]$ に対応する分散値 W_{ij} をオーナに送る.
4. サーバにより生成された分散値及び擬似乱数を受け取ったオーナは, これらを利用して (k, n) 閾値秘密分散法と同様の手段で, 秘密情報 s_i を復元する.

2.3 Herzberg の更新手法

Herzberg らの更新手法では分散値の更新を行うだけの基本方式と, 更新時または復元時に不正サーバがいる場合を想定し, 検証情報によってその正当性を確認する検証手法2つが提案されている. 以下にそれらを示す.

2.3.1 基本方式

n 台のサーバに 2.1 の手法で分散値が保存されているとして, 以下にその更新手順を示す. ただし, 各サーバは互

いに安全な暗号通信を行うことができるとする.

1. サーバ P_i は $k-1$ 個の乱数 $b_{ij} (j = 1, 2, \dots, k-1)$ を生成し, 式(7)を満たす $k-1$ 次多項式 $b_i(x)$ を作る.

$$b_i(x) = b_{i1}x^1 + b_{i2}x^2 + \dots + b_{ik-1}x^{k-1} \quad (8)$$

$$b_i(0) = 0 \quad (9)$$

2. サーバ P_i は他のすべてのサーバ P_j の $ID x_j$ を x に入力した $u_{ij} = b_i(x_j)$ を生成し, 暗号化して安全に送る.
3. 暗号化された u_{ij} を受け取ったサーバ P_j は u_{ij} を復号し自身の分散値に加える. ($W_j^{(t)}$ は時刻 t における分散値を表し, t は分散値の更新が行われる度に +1 される)

$$W_j^{(t)} = W_j^{(t-1)} + (u_{1j} + u_{2j} + \dots + u_{nj}) \quad (10)$$

以上によって, すべてのサーバが正しく上記手順を行う場合, 分散値の更新が正しく行われる.

2.3.2 更新時での不正に対する検証方式

2.3.1 において, あるサーバが式(9)を満足しない $k-1$ 次方程式を使って u_{ij} を作成し, それを配布した場合, 正しい分散値の更新は行われぬ. すなわち, どの k 台のサーバの分散値を集めても正しい秘密情報は復元されない (不正を行ったサーバはその差分値を知るため, そのサーバのみ正しく秘密情報を復元できる).

よって, 式(9)を満足しない情報を送る攻撃を想定する場合は以下の手順で更新を行う.

以下において, 送信者を S , と受信者を R としたとき, ENC_R は受信者の公開鍵で暗号化することを意味し, SIG_S は送信者の秘密鍵で署名することを意味する.

1. サーバ P_i は $k-1$ 個の乱数を生成し式(11)の $k-1$ 次多項式を作る. また, 式(11)の多項式の各係数に対して式(12)を計算する.

$$b_i(x) = b_{i1}x^1 + b_{i2}x^2 + \dots + b_{ik-1}x^{k-1} \quad (11)$$

$$\varepsilon_{im} = g^{b_{im}} (m = 1, 2, \dots, k-1) \quad (12)$$

2. サーバ P_i は他のすべてのサーバ P_j に対して $u_{ij} = b_i(j)$ と $e_{ij} = ENC_j(u_{ij})$ を計算する.
3. サーバ P_i は $VSS_i^{(t)} = (i, t, \varepsilon_{im}, e_{ij})$ と $SIG_i(VSS_i^{(t)})$ をブロードキャストする. ($VSS_i^{(t)}$ は時間 t における VSS_i)
4. サーバ P_i は送られてきた情報 e_{ij} を復号し, 以下の式で検証する.

$$g^{u_{ji}} = (\varepsilon_{j1})^i (\varepsilon_{j2})^{i^2} \dots (\varepsilon_{jk})^{i^k} \quad (13)$$

5. 正確な情報と判断したら, 送られてきた情報を保持し, 検証が通ったことを他のサーバにブロードキャストする.
6. すべてのサーバの検証が通ったらサーバ P_i は $b_j(i)$ を自身の分散値に加える.
7. 5. で検証が通らなかった際には, 不正サーバから送ら

れる b_i を使用しない。また、不正サーバの存在を他のサーバにブロードキャストする。

2.3.3 復元時での不正に対する検証方式

さらに、復元時に不正な分散値を提出するサーバに対して行う検証法について説明する。

1. 分散値の生成時、分散値生成者は検証鍵 $y_i^{(t)} = g^{W_i^{(t)}}$ をブロードキャストする。
2. 分散値の更新時、2.3.1の3.または、2.3.2の6において、各サーバは以下の式によって検証鍵の更新を行い、ブロードキャストする。

$$y_i^{(t)} = y_i^{(t-1)} \times (g^{u_{1i}} \times g^{u_{2i}} \times \dots \times g^{u_{ni}}) \quad (14)$$
3. 秘密情報の復元時、復元者は集めた分散値を検証鍵で検証を行うことで分散値が正しいかものか判別する。

なお、安全な掲示板などを準備できる場合、ブロードキャストは掲示板での公開とすることもできる。

3. 提案方式

3.1 提案方式 1

2.2に示す非対称秘密分散方式を対象として、秘密情報のオーナーが鍵サーバとなる場合における分散値の更新を考える。また、オーナーによる秘匿演算・復元の管理を実現するためデータサーバ台数は $n-l < k$ を満たすものとする。データサーバの数を t とすると、全サーバ数は $n = l + t$ ($l < k, t < k$)となる。

非対称秘密分散法は秘密情報オーナーがその復元・利用を制御できる点が特徴である。よって、提案方式1では、その特徴を利用してオーナーが分散値の更新も制御できる手法を示す。提案方式1においても、分散値の更新を行う基本方式と不正なサーバがいる場合の検証可能方式の2つを示す。ただし、提案方式1では更新に必要な情報を送るのはオーナーのみなので、更新時での不正に対する検証方式は考えないものとする。

3.1.1 基本方式

[更新]

時刻 $t-1$ において各サーバ P_j は既に m 個の秘密情報 s_i ($i = 1, \dots, m$)に対する分散値 $W_{ij}^{(t-1)}$ を所持しているとし、時刻 t において更新を行う。

1. オーナは自身の携帯端末に新しい鍵 $key_o^{(t)}$ を設定し、新しい鍵 $key_o^{(t)}$ と以前の鍵 $key_o^{(t-1)}$ より $k-1$ 個の更新用の鍵 $key2_j^{(t)}$ ($j = n-k+2, \dots, n$), $key2_j^{(t-1)}$ ($j = n-k+2, \dots, n$)を生成する。

$$key2^{(t)} = Enc(t, key_o^{(t)}) \quad (15)$$

$$key2^{(t-1)} = Enc(t-1, key_o^{(t-1)}) \quad (16)$$

$$key2_j^{(t)} = Enc(j, key2^{(t)}) \quad (j = n-k+2, \dots, n) \quad (17)$$

$$key2_j^{(t-1)} = Enc(j, key2^{(t-1)}) \quad (j = n-k+2, \dots, n) \quad (18)$$

2. オーナはデータサーバ $P_{l+1} \sim P_n$ に対応する更新用の鍵 $key2_j^{(t)}$ ($j = l+1, \dots, n$)を送る。
3. 全てのデータサーバは更新用の鍵より m 個の差分 $u_{ij}^{(t)}, u_{ij}^{(t-1)}$ を生成する。

$$u_{ij}^{(t)} = Enc(dID[s_i], key2_j^{(t)}) \quad (i = 1, \dots, m) \quad (19)$$

$$u_{ij}^{(t-1)} = Enc(dID[s_i], key2_j^{(t-1)}) \quad (i = 1, \dots, m) \quad (20)$$

ここで、新しい分散値 $W_i^{(t)}$, $W_i^{(t-1)}$ と差分 $u_{ij}^{(t)}, u_{ij}^{(t-1)}$ は式(20), (21), (22), (23)のように表されるものとする。

$a_{i1}^{(t)}, \dots, a_{ik-1}^{(t)}$ と $a_{i1}^{(t-1)}, \dots, a_{ik-1}^{(t-1)}$ は分散式の係数である。

$$W_{ij}^{(t)} = s_i + a_{i1}^{(t)} x_j + a_{i2}^{(t)} x_j^2 + \dots + a_{ik-1}^{(t)} x_j^{k-1} \quad (21)$$

$$W_{ij}^{(t-1)} = s_i + a_{i1}^{(t-1)} x_j + a_{i2}^{(t-1)} x_j^2 + \dots + a_{ik-1}^{(t-1)} x_j^{k-1} \quad (22)$$

$$u_{ij}^{(t)} = (a_{i1}^{(t)} - a_{i1}^{(0)}) x_j + (a_{i2}^{(t)} - a_{i2}^{(0)}) x_j^2 + \dots + (a_{ik-1}^{(t)} - a_{ik-1}^{(0)}) x_j^{k-1} \quad (23)$$

$$u_{ij}^{(t-1)} = (a_{i1}^{(t-1)} - a_{i1}^{(0)}) x_j + (a_{i2}^{(t-1)} - a_{i2}^{(0)}) x_j^2 + \dots + (a_{ik-1}^{(t-1)} - a_{ik-1}^{(0)}) x_j^{k-1} \quad (24)$$

3. 全てのデータサーバは以下の式より新たな分散値 $W_{ij}^{(t)}$ を得る。

$$W_{ij}^{(t)} = W_{ij}^{(t-1)} - u_{ij}^{(t-1)} + u_{ij}^{(t)} \quad (25)$$
4. 更新後、全てのデータサーバは以前の分散値 $W_{ij}^{(t-1)}$ と更新用の鍵 $key2_j^{(t)}, key2_j^{(t-1)}$ を消去し、オーナーは以前の鍵 $key_o^{(t-1)}$ を消去する。

最終的に、オーナーは初期鍵 key_o と新しい鍵 $key_o^{(t)}$ を持つ。

[復元]

1. オーナは初期鍵 key_o と自身の暗号装置より l 個の鍵 $Eid(j)$ ($j = 1, \dots, l$)を式(14)より生成する。
2. オーナは l 個の鍵 $Eid(j)$ ($j = 1, \dots, l$)と自身の暗号装置より l 個の疑似乱数 q_{ij} ($j = 1, \dots, l$)を生成する。

$$q_{ij} = Enc(dID[s_i], Eid(j)) \quad (j = 1, \dots, l) \quad (26)$$
3. オーナは $k-l$ 個のサーバ P_j ($j = l+1, \dots, k$)を選択する。
4. サーバ P_j ($j = l+1, \dots, k$)はオーナーに更新後の分散値 $W_{ij}^{(t)}$ ($j = l+1, \dots, k$)を暗号化し安全に送る。
5. オーナは新しい鍵 $key_o^{(t)}$ と自身の暗号装置より $k-1$ 個の更新用の鍵 $key2_j^{(t)}$ ($j = n-k+2, \dots, n$)を式(15), (17)より生成する。

6. オーナは $k-1$ 個の更新用の鍵 $key2_j^{(t)} (j = n-k+2, \dots, n)$ と $dID[s_i]$ と自身の持つ暗号装置より差分 $k-1$ 個の u_{ij} を生成する。

$$u_{ij} = Enc(dID[s_i], key2_j^{(t)}) (j = n-k+2, \dots, n) \quad (27)$$

7. オーナは $k-1$ 個の差分情報から式(9)を連立させて、時刻 t と時刻 $t-1$ の分散式の各係数の変化量 $(a_{i1}^{(t)} - a_{i1}^{(t-1)}), \dots, (a_{ik-1}^{(t)} - a_{ik-1}^{(t-1)})$ を計算し、それらから疑似乱数 $q_{ij} (j = 1, \dots, l)$ の差分 $u_{ij} (j = 1, \dots, l)$ を計算する。
8. 疑似乱数 $q_{ij} (j = 1, \dots, l)$ と差分 $u_{ij} (j = 1, \dots, l)$ を加算することにより更新後のオーナの分散値 $W_{ij}^{(t)} (j = 1, \dots, l)$ を得る。

$$W_{ij}^{(t)} = q_{ij} + u_{ij} \quad (28)$$

l 個のオーナの分散値 $W_{ij}^{(t)} (j = 1, \dots, l)$ とサーバ $P_j (j = l+1, \dots, k)$ からの分散値 $W_{ij}^{(t)} (j = l+1, \dots, k)$ より秘密情報 s_i を復元する。

3.1.2 復元時での不正に対する検証方式

Herzberg らの更新手法では離散対数問題に基づく検証を行うため、べき乗演算を必要とした。それに対して、本提案では非対称秘密分散法の特徴を利用してべき乗演算を用いない手法を示す。すなわち、オーナが生成する分散値は正しいということを前提とする検証方式である。

オーナは全てのサーバからそれぞれ1個ずつの分散値をもらい、自分の鍵より生成した l 個の分散値と組合せて復元した場合、その分散値の1つでも違えば必ず復元した秘密情報は異なるということを利用する。

1. オーナは自身の鍵 key_o と自身の暗号装置より l 個の疑似乱数 $q_j (j = 1, \dots, l)$ を生成する。
2. オーナは全てのサーバからそれぞれ1個ずつの分散値を得る。
3. オーナは自身の鍵から生成した l 個の分散値と全てのサーバから得た分散値のうち $k-l$ 個を用いて tC_{k-l} 個のパターンにおいて秘密情報を復元する。
4. オーナは復元した秘密情報を比較し、一致しているかを検討する。
5. 復元に用いた分散値から復元した秘密情報が他と異なっている場合、その中に嘘の分散値を送っているサーバがあるため、復元した秘密情報は不正であるとする。

3.2 提案方式2

3.2.1 基本方式

従来方式である Herzberg らによる更新法では、更新に用いる式 b_{ij} が定数項を0とする $k-1$ 次式であるため、 $k-1$ 台のサーバを攻撃されると更新情報が漏洩してしまうという問題があった。そこで、更新に用いる多項式 $b_i(x)$ を

(定数項) $\neq 0$ である k 次多項式とすることを考える。更新される分散値 $W^{(t-1)}$ は秘匿化分散値 $W'^{(t)}$ となり、分散値 $W^{(t)}$ に疑似乱数 $r^{(t)}$ を加算した形 $W'^{(t)} = W^{(t)} + r^{(t)}$ で表される。また、各サーバは秘匿化を解除するための疑似乱数 $r^{(t)}$ の分散値 $R^{(t)}$ を同時に生成する。復元の際は秘匿化分散値 W' と分散値 R をそれぞれ k 個集めることで秘密情報 s を復元することができる。以上のように、 k 閾値における分散値の更新を実現した。

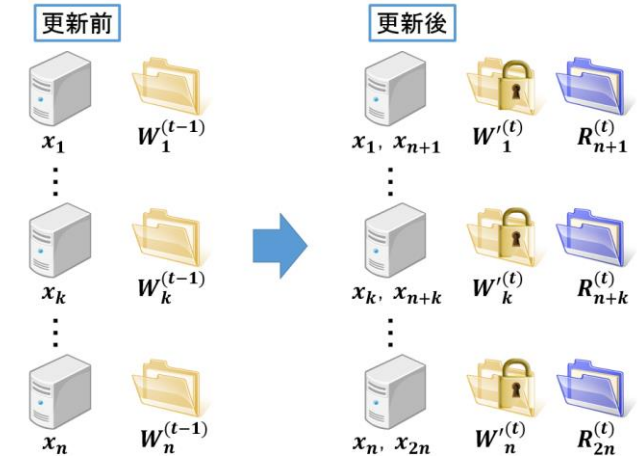


図1 更新

[更新]

時刻 $t-1$ において各サーバ P_j は既に秘密情報 $s \in \mathbf{Z}/p\mathbf{Z}$ (p は素数)に対する秘匿化分散値 $W_i'^{(t-1)}$ と疑似乱数 $r^{(t-1)}$ の分散値 $R_i^{(t-1)}$ を所持しているとする。また、1度目の更新の場合、以前の疑似乱数 $r^{(t-1)}$ は存在しないため、 $r^{(t-1)} = 0$ 、 $W_i'^{(t-1)} = W_i^{(t-1)}$ となる。

ここで、提案方式における計算は基本的に $\mathbf{Z}/p\mathbf{Z}$ 上で行われるものとする。時刻 t において以下の手順で更新を行う。

4. n 台のサーバ P_i の各々サーバID1として $x_i (i = 1, 2, \dots, n)$ を割り当て、さらに1つのサーバID2として $x_i (i = n+1, n+2, \dots, 2n)$ を各々割り当てる。
5. サーバ P_i は $2k-1$ 個の乱数 $\gamma_i^{(t)}$ 、 $b_{ij}^{(t)} (j = 1, 2, \dots, k-1)$ 、 $c_{ij}^{(t)} (j = 1, 2, \dots, k-1)$ を生成し、 k 次多項式 $b_i'^{(t)}(x)$ 、 $c_i'^{(t)}(x)$ を作る。ただし、 $b_i'^{(t)}(x)$ に用いる x はID1とし、 $c_i'^{(t)}(x)$ に用いる x はID2とする。

$$b_i'^{(t)}(x) = \gamma_i^{(t)} + b_{i1}^{(t)}x + b_{i2}^{(t)}x^2 + \dots + b_{ik-1}^{(t)}x^{k-1} \quad (29)$$

$$c_i'^{(t)}(x) = \gamma_i^{(t)} + c_{i1}^{(t)}x + c_{i2}^{(t)}x^2 + \dots + c_{ik-1}^{(t)}x^{k-1} \quad (30)$$

6. サーバ P_i は他のすべてのサーバ P_j に $u_{ij}^{(t)} = b_i'^{(t)}(x_j)$ 、 $v_{in+j}^{(t)} = c_i'^{(t)}(x_{n+j})$ を暗号を用いて安全に送る。
7. 暗号化された $u_{ij}^{(t)}$ 、 $v_{in+j}^{(t)}$ を受け取ったサーバ P_j は、 $u_{ij}^{(t)}$ 、 $v_{in+j}^{(t)}$ を復号し全て加算することにより、分散値 $U_j^{(t)}$ 、 $V_{n+j}^{(t)}$ を計算する。

$$U_j^{(t)} = u_{1j}^{(t)} + u_{2j}^{(t)} + \dots + u_{nj}^{(t)} \quad (31)$$

$$V_{n+j}^{(t)} = v_{1n+j}^{(t)} + v_{2n+j}^{(t)} + \dots + v_{mn+j}^{(t)} \quad (32)$$

ここで、分散値 $U_j^{(t)}$ 、 $V_{n+j}^{(t)}$ は式(5)、(6)のように表され

るものとする. $B_1^{(t)}, \dots, B_{k-1}^{(t)}, C_1^{(t)}, \dots, C_{k-1}^{(t)}$ は分散式の係数である.

$$U_j^{(t)} = (r^{(t)} - r^{(t-1)}) + B_1^{(t)} x_j + B_2^{(t)} x_j^2 + \dots + B_{k-1}^{(t)} x_j^{k-1} \quad (33)$$

$$V_{n+j}^{(t)} = (r^{(t)} - r^{(t-1)}) + C_1^{(t)} x_{n+j} + C_2^{(t)} x_{n+j}^2 + \dots + C_{k-1}^{(t)} x_{n+j}^{k-1} \quad (34)$$

$$r^{(t)} = r^{(t-1)} + \gamma_1^{(t)} + \gamma_2^{(t)} + \dots + \gamma_n^{(t)} \quad (35)$$

$$B_i^{(t)} = b_{1i}^{(t)} + b_{2i}^{(t)} + \dots + b_{ni}^{(t)} \quad (i = 1, \dots, k-1) \quad (36)$$

$$C_i^{(t)} = c_{1i}^{(t)} + c_{2i}^{(t)} + \dots + c_{ni}^{(t)} \quad (i = 1, \dots, k-1) \quad (37)$$

8. サーバ P_j は前の更新された秘匿化分散値 $W_j^{(t-1)}$ と分散値 $U_j^{(t)}$, 分散値 $R_j^{(t-1)}$ と分散値 $V_j^{(t)}$ を加算することで,

更新後の秘匿化分散値 $W_j^{(t)}$ と分散値 $R_j^{(t)}$ を生成する.

$$W_j^{(t)} = W_j^{(t-1)} + U_j^{(t)} \quad (38)$$

$$R_{n+j}^{(t)} = R_{n+j}^{(t-1)} + V_{n+j}^{(t)} \quad (39)$$

ここで, 前の更新された秘匿化分散値 $W_j^{(t-1)}$ 新しい秘匿化分散値 $W_j^{(t)}$, 前の更新された分散値 $R_{n+j}^{(t-1)}$, 新しい分散値 $R_{n+j}^{(t)}$ は式(10), (11), (12)のように表されるものとする. $a_1^{(t-1)}, \dots, a_{k-1}^{(t-1)}, d_1^{(t-1)}, \dots, d_{k-1}^{(t-1)}$ は分散式の係数である.

$$W_j^{(t-1)} = (s + r^{(t-1)}) + a_1^{(t-1)} x_j + a_2^{(t-1)} x_j^2 + \dots + a_{k-1}^{(t-1)} x_j^{k-1} \quad (40)$$

$$W_j^{(t)} = (s + r^{(t)}) + (a_1^{(t-1)} + B_1^{(t)}) x_j + \dots + (a_{k-1}^{(t-1)} + B_{k-1}^{(t)}) x_j^{k-1} \quad (41)$$

$$R_{n+j}^{(t-1)} = r^{(t-1)} + d_2^{(t-1)} x_{n+j} + d_2^{(t-1)} x_{n+j}^2 + \dots + d_{k-1}^{(t-1)} x_{n+j}^{k-1} \quad (42)$$

$$R_{n+j}^{(t)} = r^{(t)} + (d_1^{(t-1)} + C_1^{(t)}) x_{n+j} + (d_2^{(t-1)} + C_2^{(t)}) x_{n+j}^2 + \dots + (d_{k-1}^{(t-1)} + C_{k-1}^{(t)}) x_{n+j}^{k-1} \quad (43)$$

以上によって, すべてのサーバが正しく上記手順を行う場合, 更新が正しく行われる.

時刻 t で更新した後, 秘密情報 s を復元する方法を以下に示す.

[復元]

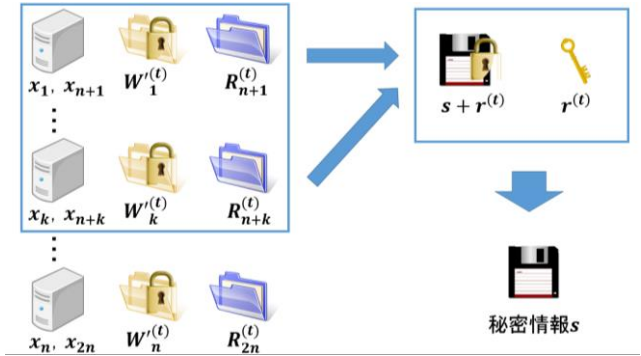


図3 更新後の復元手法

1. 復元者は, k 台のサーバ $P_i (i = 1, \dots, k)$ を選択する.
2. サーバ $P_i (i = 1, \dots, k)$ は, 秘匿化分散値 $W_i^{(t)}$ と分散値 $R_{n+i}^{(t)}$ を復元者に送る.
3. 復元者は k 個の秘匿化分散値 $W_i^{(t)}$ より, 秘密情報 s と擬似乱数 $r^{(t)}$ の和 $s + r^{(t)}$ を復元する. 更に, k 個の分散値 $R_{n+i}^{(t)}$ より, 擬似乱数 $r^{(t)}$ を復元する.

復元者は, $s + r^{(t)}$ から $r^{(t)}$ を引くことで秘密情報 s を復元する.

4. 通信量評価

提案方式1では, オーナが t 台のデータサーバに2つの更新用の鍵のみを送信する. 必要な通信は秘密情報の数にかかわらず $2(n-l)$ 回となる.

表1 通信回数比較 ($n = 4, k = 3, l = 2, m = 1000$)

Herzberg の更新手法 (基本方式)	提案方式
12000 回	4 回

表1より, 通信回数の大幅な削減が可能である.

5. 更新時・更新後の安全性

提案方式1と提案方式2における更新時・更新後の安全性について検討する.

5.1 提案方式1

オーナが携帯端末をなくすまたは盗難された場合, パスワードや生体認証などを用いて厳重に管理されていたとしても, オーナの鍵の情報が漏洩する可能性がある. したがって, 「攻撃者はオーナの鍵と前の更新用の鍵を知っている」という前提で, オーナが新しい更新用の鍵を設定すれば, 攻撃者が全データサーバを解析しても秘密情報が漏れないことを示す.

攻撃者は, オーナの鍵 $Eid(j) (j = 1, \dots, l)$ と前の更新用の鍵 $key2_j^{(t-1)} (j = n - k + 2, \dots, n)$ より, l 個の擬似乱数 $q_{ij} (j = 1, \dots, l)$ と $k-1$ 個の差分 $u_{ij}^{(t-1)} (j = n - k + 2, \dots, n)$ を得ることができる.

$$q_{ij} = Enc(dID[s_i], Eid(j)) \quad (j = 1, \dots, l) \quad (44)$$

$$u_{ij}^{(t-1)} = \text{Enc}\left(dID[s_i], \text{key}2_j^{(t-1)}\right) \quad (j = n - k + 2, \dots, n) \quad (45)$$

攻撃者は、 $k-1$ 個の差分 $u_{ij}^{(t-1)}$ ($j = n - k + 2, \dots, n$)より、差分の式を決定することができる。

$$u_{ij}^{(t-1)} = (a_{i1}^{(t-1)} - a_{i1}^{(t-2)})x_j + (a_{i2}^{(t-1)} - a_{i2}^{(t-2)})x_j^2 + \dots + (a_{ik-1}^{(t-1)} - a_{ik-1}^{(t-2)})x_j^{k-1} \quad (46)$$

攻撃者は、オーナの $IDx_i (i = 1, \dots, l)$ から式(23)より差分 $u_{ij}^{(t-1)}$ ($j = 1, \dots, l$)を得ることができる。

疑似乱数 $q_{ij} (j = 1, \dots, l)$ と差分 $u_{ij}^{(t-1)}$ ($j = 1, \dots, l$)を加算することにより前の更新された分散値 $W_{ij}^{(t-1)}$ ($j = 1, \dots, l$)を得ることができる。

$$W_{ij}^{(t-1)} = q_{ij} + u_{ij}^{(t-1)} \quad (j = 1, \dots, l) \quad (47)$$

以下において更新時と更新後の安全性について考える。

更新前

1. 攻撃者がオーナの鍵 $Eid(j) (j = 1, \dots, l)$ と前の更新用の鍵 $\text{key}2_j^{(t-1)} (j = n - k + 2, \dots, n)$ を知っている場合、攻撃者は $k-l$ 台のデータサーバを攻撃すると、 $k-l$ 個の前の更新された分散値が漏洩する。合計で k 個の前の更新された分散値より秘密情報を決定することができるので以下の式が成立する。

$$H(s_i | W_{i1}^{(t-1)}, W_{i2}^{(t-1)}, \dots, W_{ik}^{(t-1)}) = 0 \quad (48)$$

2. 攻撃者がオーナの鍵 $Eid(j) (j = 1, \dots, l)$ と前の更新用の鍵 $\text{key}2_j^{(t-1)} (j = n - k + 2, \dots, n)$ を知らない場合、攻撃者は $k-l$ を超える $k-1$ 台までのデータサーバを攻撃しても、 k 個の分散値はわからないので秘密情報について以下の式が成立する。

$$H(s_i | W_{i1}^{(t-1)}, W_{i2}^{(t-1)}, \dots, W_{ik-1}^{(t-1)}) = H(s_i) \quad (49)$$

更新時

3. 攻撃者は、更新時に $k-1$ 台のデータサーバを攻撃すると新しい更新用の鍵 $\text{key}2_j^{(t)} (j = n - k + 2, \dots, n)$ より $k-1$ 個の差分が漏洩する。 $k-1$ 個の差分情報より差分の式の係数を全て決定することができるので以下の式が成立する。

$$H(a_{im}^{(t)} - a_{im}^{(t-1)} | u_{i1}^{(t)}, u_{i2}^{(t)}, \dots, u_{ik-1}^{(t)}) = 0 \quad (m = 1, \dots, k-1) \quad (50)$$

4. 攻撃者は、更新時に $k-2$ 台のデータサーバを攻撃すると新しい更新用の鍵 $\text{key}2_j^{(t-1)} (j = n - k + 2, \dots, n-1)$ より $k-2$ 個の差分情報が漏洩する。 $k-2$ 個の差分情報より差分の式の係数について何もわからないので以下の式が成立する。

$$H(a_{im}^{(t)} - a_{im}^{(t-1)} | u_{i1}^{(t)}, u_{i2}^{(t)}, \dots, u_{ik-2}^{(t)}) = H(a_{im}^{(t)} - a_{im}^{(t-1)}) \quad (m = 1, \dots, k-2) \quad (51)$$

更新後

更新時に攻撃されるデータサーバが $k-2$ 台以下の場合の更新後の安全性について考える。攻撃者は新しい差分の式を決定することができないため、差分の式にオーナの $IDx_i (i = 1, \dots, l)$ を代入することによって求める $u_{ij}^{(t)} (j = 1, \dots, l)$ を得ることはできない。 $W_{ij}^{(t)} = q_{ij} + u_{ij}^{(t)} (j = 1, \dots, l)$ より、 $q_{ij} (j = 1, \dots, l)$ から $W_{ij}^{(t)} (j = 1, \dots, l)$ に関する情報は何もわからない。

$$H(W_{ij}^{(t)} | q_{ij}) = H(W_{ij}^{(t)}) \quad (j = 1, \dots, l) \quad (52)$$

よって、以下が言える。

5. 攻撃者は、更新後に $k-l$ を超える $k-1$ 台のサーバを攻撃しても、合計で $k-1$ 個の更新された分散値がわからないので、秘密情報について以下の式が成立する。

$$H(s_i | W_{i1}^{(t-1)}, W_{i2}^{(t-1)}, \dots, W_{ik-1}^{(t-1)}) = H(s_i) \quad (53)$$

以上より、携帯端末をなくして分散値を更新しないと1. より $k-l$ 台のデータサーバが攻撃されると秘密情報が漏洩する。それに対して、3. ,4. より $t < k-1$ として提案方式による更新を行えば、更新時に全データサーバを盗聴されても更新後の分散値はわからない。よって、 $l > 1$ とすれば、5. より2. の状態に戻り、更新によって安全性が回復することが言える。

5.2 提案方式2

前提として、時刻 $t-1$ において攻撃者は $k-1$ 個の秘匿化分散値 $W_i^{(t-1)}$ と分散値 $R_{n+i}^{(t)} (i = 1, \dots, k-1)$ を知っているものとする。そして、時刻 t で分散値の更新を行うことにより、安全性が k 閾値に回復することを示す。また、卒論の方式では、更新時に $k-1$ 台のサーバを更新された場合、差分の式が漏洩するという問題があった。本手法では、 $k-1$ 台のサーバを攻撃されても更新に関する式を決定することができないことを示す。

更新前

6. 攻撃者が $k-1$ 個の前の更新された秘匿化分散値 $W_i^{(t-1)}$ と分散値 $R_{n+i}^{(t-1)}$ を知っている場合、攻撃者は1台のデータサーバを攻撃すると、1個の前の更新された分散値 $W_i^{(t-1)}$ と分散値 $R_{n+i}^{(t-1)}$ が漏洩する。合計で k 個の前の更新された分散値より秘密情報を決定することができる。

$$H(s | W_1^{(t-1)}, \dots, W_k^{(t-1)}, R_{n+1}^{(t-1)}, \dots, R_{n+k}^{(t-1)}) = 0 \quad (54)$$

$$H(s + r | W_1^{(t-1)}, \dots, W_k^{(t-1)}) = 0 \quad (55)$$

$$H(r | R_{n+1}^{(t-1)}, \dots, R_{n+k}^{(t-1)}) = 0 \quad (56)$$

7. 攻撃者が秘匿化分散値 $W_i^{(t-1)}$ と分散値 $R_{n+i}^{(t-1)}$ を 1 個も知らない場合、攻撃者は $k-1$ 台までのデータサーバを攻撃しても、 k 個の分散値はわからないので秘密情報について以下の式が成立する。

$$H(s | W_1^{(t-1)}, \dots, W_{k-1}^{(t-1)}, R_{n+1}^{(t-1)}, \dots, R_{n+k-1}^{(t-1)}) = H(s) \quad (57)$$

$$H(s + r | W_1^{(t-1)}, \dots, W_{k-1}^{(t-1)}) = H(s + r) \quad (58)$$

$$H(r | R_{n+1}^{(t-1)}, \dots, R_{n+k-1}^{(t-1)}) = H(r) \quad (59)$$

更新時

攻撃者が更新時に $k-1$ 台のデータサーバを攻撃すると、前の更新された秘匿化分散値 $W_i^{(t-1)}$ と分散値 $R_{n+i}^{(t-1)}$ と今回の更新の際に利用する分散値 $U_i^{(t)}$ 、 $V_i^{(t)}$ が漏洩する。

$$W_i^{(t-1)} = (s + r^{(t-1)}) + a_1^{(t-1)} x_i + a_2^{(t-1)} x_i^2 + \dots + a_{k-1}^{(t-1)} x_i^{k-1} \quad (60)$$

$$R_{n+i}^{(t-1)} = r^{(t-1)} + d_1^{(t-1)} x_{n+i} + d_2^{(t-1)} x_{n+i}^2 + \dots + d_{k-1}^{(t-1)} x_{n+i}^{k-1} \quad (61)$$

$$U_i^{(t)} = (r^{(t)} - r^{(t-1)}) + B_1^{(t)} x_i + B_2^{(t)} x_i^2 + \dots + B_{k-1}^{(t)} x_i^{k-1} \quad (62)$$

$$V_{n+i}^{(t)} = (r^{(t)} - r^{(t-1)}) + C_1^{(t)} x_{n+i} + C_2^{(t)} x_{n+i}^2 + \dots + C_{k-1}^{(t)} x_{n+i}^{k-1} \quad (63)$$

8. 攻撃者は、 $k-1$ 個の $W_i^{(t-1)}$ 、 $R_{n+i}^{(t-1)}$ 、 $U_i^{(t)}$ 、 $V_{n+i}^{(t)}$ を得ることができるが、分散値の式 $U_i^{(t)}$ 、 $V_{n+i}^{(t)}$ の係数を決定することはできない。

$$H(B_i^{(t)} | W_1^{(t-1)}, \dots, W_{k-1}^{(t-1)}, R_1^{(t-1)}, \dots, R_{k-1}^{(t-1)}, U_1^{(t)}, \dots, V_1^{(t)}, \dots, V_{k-1}^{(t)}) = H(s) \quad (i = 1, \dots, k-1) \quad (64)$$

$$H(C_i^{(t)} | W_1^{(t-1)}, \dots, W_{k-1}^{(t-1)}, R_1^{(t-1)}, \dots, R_{k-1}^{(t-1)}, U_1^{(t)}, \dots, V_1^{(t)}, \dots, V_{k-1}^{(t)}) = H(s) \quad (i = 0, 2, \dots, k-1) \quad (65)$$

更新後

更新時にデータサーバが攻撃されないとすると、分散値 $U_i^{(t)}$ 、 $V_i^{(t)}$ を得ることができないため、更新前の秘匿化分散値 $W_i^{(t-1)}$ と分散値 $R_i^{(t-1)}$ から、更新後の秘匿化分散値 $W_i^{(t)}$ と分散値 $R_i^{(t)}$ に関する情報は一切得られない。

$$H(W_i^{(t)} | W_i^{(t-1)}) = H(W_i^{(t)}) \quad (i = 1, \dots, k-1) \quad (66)$$

$$H(R_{n+i}^{(t)} | R_{n+i}^{(t-1)}) = H(R_{n+i}^{(t)}) \quad (67)$$

$$(i = 1, \dots, k-1)$$

よって、以下が言える。

9. 攻撃者は、更新後に $k-1$ 台以下のサーバを攻撃しても、秘密情報に関する情報は一切得られない。

$$H(s | W_1^{(t)}, \dots, W_{k-1}^{(t)}, R_{n+1}^{(t)}, \dots, R_{n+k-1}^{(t)}) = H(s) \quad (68)$$

以上より、 $k-1$ 個の秘匿化分散値 $W_i^{(t-1)}$ と分散値 $R_{n+i}^{(t-1)}$ ($i = 1, \dots, k-1$) を知っている前提で分散値を更新しないと、1. より 1 台のデータサーバが攻撃されると秘密情報が漏洩する。また、更新時に 4. より $k-1$ 台のサーバを攻撃しても分散値 $U_i^{(t)}$ 、 $V_{n+i}^{(t)}$ の式を決定することはできない。更新時にデータサーバが攻撃されないとすると、更新後は 4. より 2. の状態に戻り、更新によって安全性が回復することが言える。

6. まとめ

本論文では、秘密情報を復元せずに分散情報を復元する手法を 2 通り示した。プロアクティブ秘密分散法の問題点として、通信量が多いことと、更新の閾値が $k-1$ となってしまうことが挙げられる。提案方式 1 では通信量の大幅な削減を実現し、提案方式 2 では更新の閾値を k とすることを実現した。

参考文献

- [1] 菊池尚也, 金井敦, 谷本茂明, 佐藤周行 “秘密分散を用いた複数クラウドのデータ管理方式” SCIS2014 Jan. 2014
- [2] A. Shamir. “How to share a secret.” Communications of the ACM, 22(11), 1979, pp. 612-613
- [3] Satoshi Takahashi, Hyunho Kang, Keiichi Iwamura. “Asymmetric secret sharing scheme suitable for cloud systems,” 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), Las Vegas, Jan. 2014, pp. 798-804
- [4] Herzberg, Amir et al. “Proactive secret sharing or: How to cope with perpetual leakage,” Advances in Cryptology CRYPTO '95. Springer Berlin Heidelberg, 1995, pp. 339-352
- [5] 古田英之, 須賀祐治, 岩村恵市 “秘密分散システムにおける分散データの更新手法” 2014-CSEC-65 May. 2014
- [6] 高橋慧, 岩村恵市 “非対称秘密分散法を用いたアプリケーションの検討” 2013-CSEC-62 July. 2013
- [7] P. Feldman, A practical scheme for non-interactive verifiable secret sharing. IEEE Symposium on Foundations of Computer Science, pages 427-437.
- [8] 栗原正純, 桑門秀典. “分散ストレージにおける再生成符号と秘密分散について.” 電子情報通信学会技術研究報告. IT, 情報理論 110. 363 (2011): 13-18.
- [9] Rashmi, Korlakai Vinayak, Nihar B. Shah, and P. Vijay Kumar. “Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction.” Information Theory, IEEE Transactions on 57.8 (2011): 5227-5239.