Vol.2014-CSEC-67 No.5
2014/12/5

秘密分散法における検証可能な分散情報の更新手法

神宮 武志[†1]　古田 英之[†1]　岩村 惠市[†1]

本稿では，秘密分散法の分散情報の更新手法について考える．著者らは[4]において，新しい分散情報の更新法を提案したが，従来，この問題に対してはプロアクティブ秘密分散法が提案されている．そこで，本稿では[4]の手法とプロアクティブ秘密分散法との比較を行い、[4]の手法にプロアクティブ秘密分散法と同等の機能を持たせるように拡張を行う．そのために，[4]の手法に検証鍵を追加し，更新情報の検証を可能にした．また通信量，計算量の比較を行った．

Updating method of verifiable distributed data in the secret sharing scheme

TAKESHI SHINGU[†1]　FURUTA HIDEYUKI[†1]
KEIICHI IWAMURA[†1]

We consider the update method of the distributed information of the secret sharing scheme. The authors have proposed a new method for updating distributed information in CSEC in May, but conventional proactive secret sharing scheme has been proposed for this problem. Therefore, the comparison is made with the proactive secret sharing scheme in this paper, we extend so as to have the same functions as the proactive secret sharing scheme. Add the verification key to the same functions as the proposed method of last, a comparison is made communication amount, the amount of calculation. In additions you have made changes to the security proof change is needed along with the additional.

1. はじめに

近年，クラウドコンピューティング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や装置に付属しているアプリなどを使って管理していた自身のデータを，オンライン上のサーバに保有，管理する．そのため，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り，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，短期間での導入が可能である．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必要な情報を知ろうとした際に処理の手順が増加し，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，一つの情報を複数の異なる情報に変換し，そのうちの一定数以上が集まれば元の情報が復元可能だが，その数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一定数以下ならば元の情報が復元でき，さらに一定数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．

秘密分散法の安全性を一定に保つために重要なこととして分散情報の更新がある．秘密分散法は分散情報が一定数集まらないと秘密情報の復元はできないが，攻撃者が一定数以下の分散情報を集めている場合，それを放置すれば，その攻撃者が分散情報を集め続けた場合いつか秘密情報が漏洩する危険性がある．この問題を解決するためには分散情報の更新が必要である．

秘密分散法において，分散情報を更新する自明な方法は一度秘密情報を復元し，再び分散するという方法である．しかし，この方法は誰かが秘密情報を復元しなければならないという問題が存在する．これを元々の秘密情報の

†1 東京理科大学
Tokyo University of Science

所有者が行うことは効率的でなく，システム側で行うことは安全性の観点から問題がある．そこで，秘密情報を復元せずに分散情報を更新する方法としてプロアクティブ秘密分散法が提案されている[2]．プロアクティブ秘密分散法は，秘密情報の安全性を一定に保つために，分散情報を定期的に更新するものである．$k-1$次方程式をすべてのサーバが生成し，それをお互いにブロードキャストし，自身の分散情報に加えることで分散情報の更新を行う．それゆえ，通信量や計算量が大きくなってしまうという問題点がある．

それに対して著者らは，[4]において，新しい分散情報の更新法を提案したが，プロアクティブ秘密分散法との比較が行われていなかった．そこで，本稿ではプロアクティブ秘密分散法との比較を行い，プロアクティブ秘密分散法と同等の機能を持たせるように拡張を行う．

本論文の構成を以下に示す．第 2 章において秘密分散法と，プロアクティブ秘密分散法，及びその問題点について説明する．その後第 3 章では，提案方式の説明をし，検証可能な提案方式について説明する．4 章で，本論分のまとめを行う．

2. 従来方式

2.1 (k,n)閾値秘密分散法

次の 2 つの条件を満たす秘密分散法を(k,n)閾値秘密分散法と呼ぶ．

1. 任意のk個の分散情報から，元の分散情報sを復元することができる
2. $k-1$個以下の分散情報からは，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)閾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. GF(p)の元から，異なるn個の$x_i(i=1,2,\cdots,n)$ を選び出し，サーバ ID とする．
3. GF(p)の元から，$k-1$個の乱数$a_l(l=1,2,\cdots,k-1)$ を選んで，以下の式を生成する．

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

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

[復号]

1. 復号に用いる分散情報を$W_i(i=1,2,\cdots,k)$とする．また，その分散情報に対応するユーザ ID をx_iとする．
2. 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 プロアクティブ秘密分散法

2.2.1 基本方式

n台のサーバに 2.1 の手法で分散情報が保存されているとして，以下にその更新手順を示す．

1. サーバP_iは$k-1$個の乱数$b_{ij}(j=1,2,\cdots k-1)$を生成し，式(2)の$k-1$次方程式を作る．

$$b_i(z) = b_{i1}z^1 + b_{i2}z^2 + \cdots + b_{ik}z^{k-1} \quad (2)$$

$$b_i(0) = 0 \quad (3)$$

2. サーバP_iは他のすべてのサーバP_jに$u_{ij} = b_i(j)$を送信する．
3. サーバP_jはu_{ij}を分散情報に加える．

$$W_j^{(t)} = W_j^{(t-1)} + \left(u_{1j} + u_{2j} + \cdots + u_{nj}\right) \quad (4)$$

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，分散情報の更新が正しく行われる．しかし，あるサーバが式(3)を満足しない$k-1$次式方程式を使ってu_{ij}を作成し，それを配布した場合，正しい分散情報の更新は行われない．すなわち，どのk台のサーバの分散情報を集めても正しい秘密情報は復元されない．

2.2.2 更新段階で嘘の情報を送付し秘密情報を復元できなくする攻撃の対処法

式(3)を満足しない情報を送り，正しく秘密情報を復元できなくする攻撃に対しては以下の手順で更新を行う．

以下において，*ENC* と *SIG* は送信者を *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示し，SIG_Sは送信者の秘密鍵で署名することを示す．

1. サーバP_iは$k-1$個の乱数を生成し$k-1$次方程式を作る．また，式(6)も生成する．

$$b_i(z) = b_{i1}z^1 + b_{i2}z^2 + \cdots + b_{ik}z^{k-1} \quad (5)$$

$$\varepsilon_{im} = g^{b_{im}} \quad (6)$$

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$SIG_i\left(VSS_i^{(t)}\right)$をブロードキャストする．
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．

$$g^{u_{ji}} = (\varepsilon_{j1})^{i}(\varepsilon_{j2})^{i^2}\cdots(\varepsilon_{jk})^{i^k} \quad (7)$$

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保持していた$b_j(i)$を加える．
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．または，不正サーバを管理者に報告する．

2.2.3 復元時に不正な分散情報を提出するサーバに対する対処法

さらに，復元時に不正な分散情報を提出するサーバに対して行う検証法の手順について説明する．

1. 分散情報の生成時，分散情報生成者は検証鍵$y_i^{(t)} = g^{W_i^{(t)}}$を公開する．
2. 分散情報の更新を行うとき，2.2.1 の 3.または，2.2.2 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，公開する．

$$y_i^{(t)} = y_i^{(t-1)} * (g^{u_{1i}} * g^{u_{2i}} * \cdots * g^{u_{ni}}) \quad (8)$$

3. 秘密情報の復元時，復元者は集めた分散情報を検証鍵で検証を行うことで分散情報が正しいかものか判別する．

3. 提案方式

3.1 CSEC65 方式

以下に[4]に示される第 65 回 CSEC 研究会で提案された方式の簡単な構成手順を示す．前提として，攻撃されたサーバの分散情報は変更されていないとする（この前提はプロアクティブ秘密分散法も同様）．また，各サーバは 2 章において説明した(k,n)閾値秘密分散法を用いて既に秘密分散sに対する分散値W_iを所持しているとする．

1. $k-1$台のサーバを乱数配布サーバとし，乱数配布サーバは各々新しい分散情報となるW_{new_i}を生成し，は式(9)の関係を持つとする．

$$Wnew_i = s + a_{new_1}x_i + a_{new_2}x_i^2 + \cdots + a_{new_{k-1}}x_i^{k-1} \quad (9)$$

2. 上記$k-1$台のサーバは新しい分散情報W_{new_i}と今までの分散情報W_iとの差をとり，式(10)の値を残りの$n-k+1$台のサーバに送る．

$$u_i = (a_{new_1} - a_1)x_i + (a_{new_2} - a_2)x_i^2 + \cdots + (a_{new_{k-1}} - a_{k-1})x_i^{k-1} \quad (10)$$

3. $n+k-1$台のサーバを追従サーバとし，追従サーバは送られてきた$k-1$個の差分情報から式(10)を連立させて各乱数部分の変化量$(a_{new_i} - a_i)(i = 1,2,\cdots k-1)$を計算し，それから新たな分散情報を計算する．すなわち，$n-k+1$台のサーバは得られた各乱数部分の変化量$(a_{new_i} - a_i)(i = 1,2,\cdots k-1)$と自身のサーバ ID$x_i$から式(10)のようにして自身の差分情報を計算し，元の分散情報に加えることで，新たな分散情報

$$W_{new_i} = s + a_{new_1}x_i + a_{new_2}x_i^2 + \cdots + a_{new_{k-1}}x_i^{k-1} \quad (11)$$

を得る．

4. 以前の分散情報W_iを消去して更新完了．

3.2 更新段階で嘘の情報を送付し秘密情報を復元できなくする攻撃の対処法

プロアクティブ秘密分散法で提案されている，更新段階で不正な情報を配布し秘密情報を復元できなくする攻撃に対する CSEC65 方式の安全性を検討する．

2.2.1 に示すプロアクティブ秘密分散法では，手順(1)で行う式生成時に不正サーバが$b_i(0) \neq 0$となる式を用いて更新情報を配布すると，k個の正当なサーバを用いても秘密情報が正しく復元できないという問題があり，それを解決するため 2.2.2 の式(6)から(7)までの手順を踏む必要があった．提案方式においても同様の攻撃が考えられる．2.2.1 の手法の問題は，いくつかの不正サーバからの嘘の情報により正当なサーバの分散情報更新が正しく行われないこととといえる．それに対して，CSEC65 方式は不正サーバが嘘の情報を発信しても，正当なサーバの情報更新には影響しないことを示す．

まず，$k-1$個の乱数配布サーバのうちいくつかのサーバが嘘の差分情報を送ることを考える．提案方式は追従サーバが送られた差分情報を元に全サーバ間で矛盾のない差分情報を計算するため，2.2.1 の手法のように$b(i) = 0$とならない場合は生じない．よって，正当な k 台のサーバから集めた更新分散情報から正しく復元可能となる．次に，追従サーバのうち，いくつかのサーバが異なる値を計算し保存しても，他の正当なサーバは正しい値を計算するため，k個の正当なサーバによる情報復元は正しい秘密情報を復元する．また以上より，不正サーバの組み合わせは更新情報の配布側でも計算側でも，正当なサーバがk個以上あれば問題ないことが言える．ただし不正サーバも情報更新時に申告した情報を復元時に用いるなら正しく復元できる．以上より，CSEC65 方式はそのままで嘘の更新情報に対する耐性をもっているといえる．

よって，問題となるのは不正サーバが秘密情報の復元時に更新時に申告した分散情報と異なる不正な分散情報を出す場合のみである．この問題については次節で検討する．

3.3 復元時に不正な分散情報を提出するサーバに対する対処法

2.2.1 の手法に検証鍵を追加することで，サーバが偽の分散情報を提出したかどうかを検出できる．この機能を提案方式持たせる．

1. 分散情報の生成時，すなわち，CSEC65 方式の 1.を行う以前に，分散情報生成者は全てのサーバP_iの検証鍵$y_i = g^{W_i}$を公開する．
2. 分散情報の更新時，すなわち，CSEC65 方式の 2.のときに，以下のように検証鍵の更新を行う．$k-1$個の乱数配布サーバは式(10)に対応するg^{u_i}を公開する．
3. 追従サーバは，乱数配布サーバからの差分情報u_iとg^{u_i}が正しいことを確かめ，正しければ差分情報を用いて各乱数部分の変化量$(a_{new_i} - a_i)(i = 1,2,\cdots,k-1)$を計算し，$g^{d_i} = g^{(a_{new_i} - a_i)}$を公開する．
4. 乱数配布サーバP_iは$g^{u_i} = (g^{d_1})^{x_i} \cdots (g^{d_{k-1}})^{x_i^{k-1}}$を確認し，正しければ$W_{new_i}$を新しい分散値とする．
5. 全$g^{d_i}(i = 1,2,\cdots,k-1)$が正しければ，検証鍵を以下のように更新する．

$$y_i^{(t)} = y_i^{(t-1)} * g^{u_i} \qquad (12)$$

なお，公開は全サーバのブロードキャストでもよく，この場合各サーバは全サーバの分散情報の検証鍵を所持し，各差分情報より各検証鍵の更新を行う．

3.4 従来方式との比較

比較を行うに当たって，各方式の計算量について以下を定義する．加算の処理量をa，減算の処理量をb，乗算の処理量をc，ENCの処理量をe，その復元の処理量をd，SIGの処理量をs，真性乱数作成をr，ラグランジュ補間の処理量をl，除算の処理量をfとする

前提条件として，秘密分散法は(k,n)閾値秘密分散法，全ての計算は GF(p)上で行うものとしたうえで比較を行う．

3.4.1 計算量比較

(1) **プロアクティブ秘密分散法**

1. n 台のサーバごとに$\boldsymbol{k-1}$個の乱数$\boldsymbol{b_{im}}$を作るので，
$$(\boldsymbol{k-1}) * \boldsymbol{r} * \boldsymbol{n}$$
の計算量がかかる．また，$\boldsymbol{\varepsilon_{im} = g^{b_{im}}}$を計算するにあたって，GF($p$)上での計算なので$\boldsymbol{b_{im} \le p}$ である．よって，最大の計算量は，
$$\boldsymbol{p} * (\boldsymbol{k-1}) * \boldsymbol{c} * \boldsymbol{n}$$
2. サーバP_iはP_jに対して$u_{ij} = b(j)$を計算するので
$$\{(k-1)c + (k-1)a\} * n * n$$
$b_{ij}(j)$を ENC するので
$$e * n * n$$
また，受信側は復元をするので
$$d * n * n$$
$VSS_i^{(t)}$をブロードキャストする際署名を行うので
$$s * n$$
3. 検証計算で$g^{u_{ij}} = (\varepsilon_{j1})^{i}(\varepsilon_{j2})^{i^2} \cdots (\varepsilon_{jk})^{i^k}$の計算をするので
$$\frac{\{k*(k+1)\}}{2}c + k * a$$
4. 分散情報の更新で$x_i^{(t)} = x_i^{(t-1)} + \sum_l^n u_{li}$の計算をするので
$$(n+1) * a * n$$
5. 最後に検証鍵の更新を行うが，提案方式と同様の方式であるのでここでは無視する．

以上より，加算・減算は他の処理に比べ計算量が少なく無視できるものと考え，乗算と乱数作成，ENC，DEC，SIGについて考えると，計算量は以下のように表される．

乗算：$\left\{(k-1)*n*p + (k-1)*n^2 + \frac{k*(k+1)*n*n}{2}\right\}c$

その他：$(k-1)*r*n + e*n*n + d*n*n + s*n$

(2) **提案方式**

1. $k-1$個の乱数配布サーバがそれぞれ乱数で新しい分散情報$W_i^{(t)}$をつくるので
$$(k-1) * r$$
また，$W_i^{(t)} - W_i^{(t-1)}$を計算するので
$$(k-1) * b$$
$k-1$個の乱数配布サーバはg^{u_i}を計算するので，
$$p * (k-1) * c$$
2. $n-k+1$個の追従サーバは受け取ったg^{u_i}とu_iが正しいことを確かめるため検証計算をするので
$$p * (n-k+1) * c$$
ラグランジュ補間を用いて$k-1$個の差分情報から$(a_{newi} - a_i)$を計算するので
$$(n-k+1) * l$$
また，追従サーバは$g^{u_i} = (g^{d_1})^{x_i} \cdots (g^{d_{k-1}})^{x_i^{k-1}}$を計算するので，
$$(n-k+1) * \left\{\frac{k*(k-1)}{2} + (2+p)*(k-1)\right\} * c$$
3. $u_i = \sum_m^{k-1}\left(a_i^{(t)} - a_i^{(t-1)}\right)x_i^m$を計算するので
$$(n-k+1) * \left\{\frac{k*(k-1)}{2} + 1\right\} * c$$
4. 各乱数配布サーバは差分情報u_iを受け取った後，より差分情報が正当なものであるか検証を行うので，
$$\frac{\{k*(k+1)\}}{2}c + k * a$$
5. 分散情報の更新で$W_i^{(t)} = W_i^{(t)} + u_i$の計算をするので
$$(n-k+1) * a$$
6. 最後に検証鍵の更新を行うが，提案方式と同様の方式であるのでここでは無視する．

以上より，比較的計算量のかかる部分は乗算と除算，乱数作成であると考えると，計算量は以下のように表される．
乗算：$[(n-k+1)*\{(2+k+p)*(k-1)+p+1\}]*c+p*(k-1)*c$
その他：$(n-k+1)*l+(k-1)*r$

以上より乗算回数，乱数作成回数を比較すると提案方式の方が少ないことがわかる．また，乗算のほか，プロアクティブ秘密分散ではENG，DEC，SIGの計算が必要となり，提案方式ではラグランジュ補間の計算が必要となるが，ラグランジュ補間は，ENC，DEC，SIGと比較すると，計算量は少ないといえる．以上より，提案方式のほうが計算量は少なく優れているといえる．

3.4.2 記憶容量比較

(1) **プロアクティブ秘密分散法**

サーバ 1 台あたり分散情報 1 個と検証鍵 n 個を保持している．

(2) **提案方式**

サーバ 1 台あたり分散情報 1 個と検証鍵 n 個を保持している．

秘密分散法はどちらも Shamir の(k,n)閾値秘密分散法を用いて分散を行っているので分散情報の記憶容量は同じである．また，提案方式では，プロアクティブ秘密分散で用いていた検証鍵と同等の機能を持たせているので，どちらの方式も記憶容量においては同様である

3.4.3 通信量比較

(1) **プロアクティブ秘密分散法**

プロアクティブ秘密分散ではサーバP_iがP_jに対して，
$VSS_i^{(t)}=\left(i,t,\varepsilon_{im},e_{ij}\right)(i=1,2,\cdots n,m=1,2,\cdots,k)$ と
$SIG_i(VSS_i^{(t)})(i=1,2,\cdots n)$を送信する．また，分散情報を更新段階で改ざんし秘密情報を復元できなくする攻撃に対する対策のために検証を行うが，その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際に通信を要する．

(2) **提案方式**

提案方式では$k-1$台の乱数配布サーバが$n-k+1$台の追従サーバに差分情報u_iを送信する．また，検証鍵の更新のため全サーバが他のサーバの分散情報の差分情報が必要となるため，サーバP_iはu_i $(i=1,2,\cdots n)$をブロードキャストする必要がある．

以上より，更新の際に通信が必要なデータは，プロアクティブ秘密分散では$VSS_i^{(t)}=\left(i,t,\varepsilon_{im},e_{ij}\right)$と検証鍵であるのに対して，提案方式では差分情報と検証鍵のみでよい．以上より，提案方式のほうが通信量は小さく優れているといえる．

3.5 安全性

提案方式の安全性について議論を行う．まず，攻撃者が$t(<k)$台のサーバから分散情報を得ているとするが、更新処理中もそのサーバを盗聴し続ける場合、更新情報も知られるためプロアクティブ秘密分散法においても安全でない。よって，攻撃における前提は過去にt台のサーバの分散情報を得ている攻撃者は更新処理中にはそれらのサーバを盗聴していないとする．また，更新時のサーバ間の通信情報が盗聴された場合も，プロアクティブ秘密分散法は前記攻撃者には更新情報がわかるため安全でない．よって，サーバ間の通信は暗号などにより保護されているとする．

上記前提をおいても発生する本提案方式特有の安全性に関する問題は，追従サーバが$k-1$台の乱数配布サーバの差分情報を知ったときに，それから乱数配布サーバに関する分散情報や更新分散情報についての情報が漏洩するかということである．以下，その問題に関する安全性を考える．

本提案方式では古い分散情報を新しい分散情報に置き換えるという方法で分散情報の更新を実現した．具体的な分散情報の更新方法については，$k-1$台のサーバの新しい分散情報として真性乱数を用いて生成し，その後分散式における係数を決定することで，残りの$n-k+1$台のサーバへの分散情報の更新を行った．

ここで，本提案方式の安全性を示すため，提案方式を用いて新しく更新された分散情報が元の Shamir の秘密分散法により生成された分散情報が元の Shamir の秘密分散法により生成された分散情報と同等の情報量的安全性を持つことを示す．

まず，Shamir の秘密分散法について考える．Shamir の秘密分散法では以下の分散式により分散情報の生成を行った．

$$W_i=s+a_1x_i+a_2x_i^2+\cdots+a_{k-1}x_i^{k-1} \qquad (13)$$

これより，Shamir の秘密分散法によって求められる分散情報は真性乱数を係数とした$k-1$次方程式により定まっているため，これから求められる分散情報も真性乱数と等価であるといえる．ゆえに以下の式が成り立つ．

$$H(s)-H(s|W_1,W_2,,\cdots,W_{k-1})=0 \qquad (14)$$

これに対して，提案方式により定められる分散情報について考える．まず，最初に更新する$k-1$台のサーバが持つ分散情報は真性乱数を用いて決定される．秘密情報も真性乱数であると仮定すれば，3.1 の式(9)におけるa_{new_i}も真性乱数であるため，3.1 の式(9)を用いて得られる真性乱数と等価と考えられる．より詳細には，最初に分散情報を更新する$k-1$台のサーバは秘密情報 s や自分が持つ分散情報W_iと独立に$W_{new_i}(i=1,2,\cdots,k-1)$ を定めるので以下のことがいえる．

$$H(s) = H\left(s\middle|W_{new_1}, \cdots, W_{new_{k-1}}\right) \quad (15)$$

$$H(s) = H\left(s\middle|W_{new_1}, \cdots, W_{new_{k-1}}, W_1, \cdots, W_k\right) \quad (16)$$

このとき，各サーバは従来の$W_i(i = 1,2, \cdots, k-1)$から式(10)を計算して残りの追従サーバに送り，追従サーバは乱数部分の変化量$\left(a_{new_i} - a_i\right)(i = 1,2, \cdots, k-1)$を計算するが，$a_i(i = 1,2, \cdots k-1)$は真性乱数であり，追従サーバはその値を知らないので，乱数部分の変化量から$a_{new_i}(i = 1,2, \cdots, k-1)$を得ることはできない. よって 3.1 基本方式の手順 3.によって得られる$W_{new_i}(i = k, k+1 \cdots, n)$も真性乱数と等価といえる．以上より，提案方式では Shamir の秘密分散法と同様に情報量的安全性を持っている．

4. まとめ

本稿では，秘密分散法の分散情報の更新手法である，CSEC65 方式に関してプロアクティブ秘密分散法との比較を行い，プロアクティブ秘密分散法と同等の機能を持たせるように拡張を行った．そのために，CSEC65 方式に検証鍵を追加し，計算量，記憶容量，通信量の比較を行い，プロアクティブ秘密分散法に比べて計算量，通信量が小さい方式を提案した．

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), pp.612-613 (1979)
[3]Herzberg, Amir, et al.”Proactive secret sharing or: How to cope with perpetual leakage.” Advances in Cryptology CRYPT0 ‘ 95. Springer Berlin Heidelberg, 1995. 339-352
[4] 古田英之，須賀祐治，岩村惠市“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