情報処理学会論文誌 Vol.55 No.3 1117–1125 (Mar. 2014)

秘密分散法を利用したクラウドストレージサービスにおけるモバイル機器を考慮した安全な処理委託方式

吉田 耕太[1,a)] 西村 浩二[2] 大東 俊博[2] 相原 玲二[2]

受付日 2013年6月30日, 採録日 2013年12月4日

概要：近年のモバイル機器の普及から，ネットワークを介してデータを保管するクラウドストレージサービスには強いニーズがある．これらのサービスではデータをセキュアに保管する必要があり，そのための技術として機密性と可用性を向上させることができる秘密分散法が注目を集めている．しかし，データ容量が増えてしまう秘密分散は通信帯域の乏しいモバイル機器での処理に適していない．一方，秘密分散処理を外部サーバに委託する場合，情報漏洩の危険性があることから AES などの暗号化を組み合わせる必要があり，それにともない鍵管理コストが発生してしまう．そこで本稿では，暗号化と秘密分散法を組み合わせた方式を発展させ，通信資源に制約があるモバイル機器を考慮した鍵管理が不要で安全な処理委託方式を提案し，処理時間，通信量などを評価した．

キーワード：暗号化，秘密分散法，処理委託，鍵管理

Secure Outsourcing Scheme for Mobile Devices on Cloud Storage Services Using Secret Sharing Scheme

Kouta Yoshida[1,a)] Kouji Nishimura[2] Toshihiro Ohigashi[2] Reiji Aibara[2]

Received: June 30, 2013, Accepted: December 4, 2013

Abstract: There is a big demand in the cloud storage service to store data via network from the rapid spread of mobile devices in recent years. Because there is a need to store data in safety in these services, Secret Sharing Scheme (SSS) which can improve the availability and confidentiality as a technology suitable for it has attracted attention. However, processing of secret sharing scheme which data capacity would increase is not suitable for processing by mobile devices of poor communication bandwidth. On the other hand, when outsourcing of SSS to outside server, it is necessary to combine encryption such as AES because of the risk of information leakage, and key management costs occur due to it. In this paper, we propose a secure outsourcing scheme for mobile devices using SSS and encryption method without key management. Moreover, we evaluated processing load and communication costs for the proposed scheme.

Keywords: encryption, secret sharing scheme, outsourcing, key management

1. はじめに

スマートフォンやタブレット PC などのモバイル機器の急速的な普及により，モバイル機器をビジネス分野へ活用することに注目が集まっている．紛失や盗難によるデータの紛失や情報漏洩のリスクが高いモバイル機器をビジネスで活用する場合，顧客情報や業務データなどの重要な情報（以下，秘密情報と呼ぶ）の管理には特に注意が必要である．

そのため，それらのリスクを低減できるクラウドストレージサービスには強いニーズがある．一方，クラウドのような外部のサーバへ秘密情報を保管することには，サー

[1] 広島大学大学院総合科学研究科
The Graduate School of Integrated Arts and Sciences, Hiroshima University, Higashi-Hiroshima, Hiroshima, 739–8521 Japan
[2] 広島大学情報メディア教育研究センター
Information Media Center, Hiroshima University, Higashi-Hiroshima, HIroshima, 739–8511 Japan
[a)] m120437@hiroshima-u.ac.jp

バ障害によるデータの紛失や管理者の不正アクセスなどによる情報漏洩など，セキュリティ面での不安や法律面での問題[*1]がある．一般的に情報セキュリティには機密性，完全性，可用性の 3 要素が求められるが，秘密分散法はそのうちデータの機密性と可用性を向上させることができ，法律面の問題もクリアする[*2]，クラウドストレージサービスに適した技術として注目を集めている．

代表的な秘密分散法に Shamir の (k, n) 閾値法 [1] がある．これは秘密情報を n 個の分散情報に分割し，その中から任意の k 個の分散情報を集めることで元の秘密情報を復元できる手法であり，k 個未満の分散情報からは元の秘密情報に関する情報をまったく得ることはできないという情報理論的安全性を実現する．しかし，Shamir の方式は計算負荷が非常に高く，分散情報のデータ容量が秘密情報の n 倍になるという問題がある．計算負荷の問題に対しては排他的論理和（XOR）演算のみで構成された秘密分散法 [2], [3], [4], [5] が提案されており，高速な秘密分散処理を実現している．一方，データ容量の問題に対しては (k, L, n) ランプ型閾値秘密分散法 [6] が提案されており，$k-t$（$1 \leq t \leq L-1$）以上の分散情報から秘密情報が部分的に漏洩することを許容することで，分散情報のデータ容量を n/L 倍[*3]に抑えている．近年では，秘密分散法を実用的観点から考察する研究もさかんに行われており [7], [8], [9], [10]，NRI セキュアが提供している SecureCube/SecretShare[*4]のように，秘密分散法を利用したクラウドストレージサービスとして実用化されているものもある．

秘密分散法の研究の多くはユーザが使用するクライアント端末での処理を想定している．本稿ではクライアント端末としてモバイル機器を想定しているが，モバイル機器はデスクトップ型 PC やノート PC に比べて処理能力や通信帯域に余裕がない．そのため，計算負荷の点においては XOR 演算で構成された秘密分散法を用いることで実現可能だが，データ容量の点についてはランプ型秘密分散法といえど秘密情報よりも n/L 倍大きくなってしまうため転送効率が悪い．一方で，秘密分散処理を委託する方式 [10] も提案されている．この方式ではクライアント端末の転送負荷を軽減できる．しかし，委託先のサーバを完全に信頼する必要があり，プライベートクラウドのような限られた環境でしか使えない．AES などの暗号化と組み合わせる [9] ことで処理委託するリスクを低減させる方法も考えられるが，暗号鍵の管理コストが発生してしまう．

そこで本稿では，暗号化と秘密分散法を組み合わせた方式を発展させ，通信コストが高価で，通信資源に制約があるモバイル環境での利用を考慮した鍵管理の不要な安全な処理委託方式を提案する．本方式では，ストリーム暗号と XOR 演算のみで構成された秘密分散法を用いることで，XOR 演算の特性を利用した暗号化解除処理を導入している．また，サーバの信頼性を定義したシステムモデルを想定し，提案方式の安全性と実用性について考察する．

[*1] http://www.meti.go.jp/policy/it_policy/privacy/kojin_gadelane.htm
[*2] http://www.jipdec.or.jp/archives/ecom/results/h21seika/H21results-10.pdf
[*3] k，L，n は $1 \leq L \leq k \leq n$ を満たす整数．そのため，ランプ型秘密分散法でも生成される分散情報のデータ容量は秘密情報よりも大きくなる．
[*4] http://www.nri-secure.co.jp/service/cube/secretshare.html

2. 準備

2.1 XOR 演算を用いた (k, n) 閾値秘密分散法

排他的論理和（XOR）演算を用いた (k, n) 閾値秘密分散法については，2005 年頃から藤井，多田らの研究グループ [2], [3] や栗原らの研究グループ [4], [5] が独立に方式を提案していた．その後もデータ容量の削減など，これらの方式をさらに改良した研究も数多く行われている [7], [8], [11], [12]．

本節では具体例として栗原らの $(3, n)$ 閾値秘密分散法 [4] を用いて説明する．栗原らの $(3, n)$ 閾値秘密分散法の分散過程のアルゴリズムは**表 1** のように与えられる．ここで，$||$ はデータの連結を，$\oplus$ は XOR 演算を示し，s_0 は d ビットすべて 0 で構成されているものとする．また，本節におけるアルゴリズム上の四則演算はすべて n_p を法としたもので，たとえば，演算 $a+b$ は $(a+b) \bmod n_p$ を意味する[*5]．

$n \leq n_p$ となる素数 n_p[*6]を用いて秘密情報 s を $s =$

表 1 $(3, n)$ 閾値秘密分散法の分散アルゴリズム
Table 1 Distribution algorithm of $(3, n)$-threshold scheme.

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

$s_0 \in \{0\}^d$;

//**[Step 1]**
Divide s into $(s_1, \cdots, s_{n_p-1})$ equally;

//**[Step 2]**
For$(i = 0; i < n_p - 1; i++)$
  Choose r_i^0 from $\{0,1\}^d$ uniformly at random;
For$(i = 0; i < n_p; i++)$
  Choose r_i^1 from $\{0,1\}^d$ uniformly at random;

//**[Step 3]**
For$(i = 0; i < n; i++)$ { /* NOT n_p */
  For$(j = 0; j < n_p - 1; j++)$ {
    $w_{(i,j)} = s_{i-j} \oplus r_j^0 \oplus r_{i+j}^1$;
  }
  $w_i = w_{(i,0)}||\cdots||w_{(i,n_p-2)}$;
}

Return $(w_0, \cdots, w_{n-1})$;

[*5] 添え字の計算もこの法則に則る．
[*6] XOR 演算で構成された秘密分散法の多く [2], [3], [4], [5] は，n_p が素数でない場合には復元処理が正しく実行できない（誤ってしまう場合がある）という制約がある．栗原らの方式もこの制約があり，n_p は素数として定義してアルゴリズムが与えられている．最近の研究では，素数以外の場合でも復元処理が正しく実行されるような拡張について議論されてきている [8]．

表 2 $(3,n)$ 閾値秘密分散法の分散情報の構成
Table 2 Structure of shares in $(3,n)$-threshold scheme.

w_0	$s_0 \oplus r_0^0 \oplus r_0^1 \| \| s_{n_p-1} \oplus r_1^0 \oplus r_1^1 \| \| \cdots \| \| s_0 \oplus r_{n_p-2}^0 \oplus r_{n_p-2}^1$
w_1	$s_1 \oplus r_0^0 \oplus r_1^1 \| \| s_0 \oplus r_1^0 \oplus r_2^1 \| \| \cdots \| \| s_3 \oplus r_{n_p-2}^0 \oplus r_{n_p-1}^1$
$\vdots$	$\vdots$
w_{n-1}	$s_{n-1} \oplus r_0^0 \oplus r_{n-1}^1 \| \| s_{n-2} \oplus r_1^0 \oplus r_n^1 \| \| \cdots \| \| s_{n+1} \oplus r_{n_p-2}^0 \oplus r_{n-3}^1$

$s_1||s_2||\cdots||s_{n_p-1}$ の部分秘密情報 s_i（s_i のサイズは d ビット）に分割する．

そして，d ビットの乱数 r_0^0〜$r_{n_p-2}^0$，r_0^1〜$r_{n_p-2}^1$ を保存ごとに独立に生成し，s_i，r_i^0，r_i^1 を XOR 演算することで部分分散情報 $w_{(i,j)}$ を以下のように生成する（式 (1)）．

$$w_{(i,j)} = s_{i-j} \oplus r_j^0 \oplus r_{i+j}^1 \quad (1)$$

各分散情報 w_i は**表 2** のように部分分散情報 $w_{(i,j)}$ を連結することで構成される．

一方，復元過程のアルゴリズムは**表 3** のように与えられる．復元過程では分散情報 w_i を部分分散情報 $w_{(i,j)}$ へと分割し，各 $w_{(i,j)}$ を XOR 演算することで中間情報 u，v を経て部分秘密情報 s_i に復元する．最後に，部分秘密情報 s_i を連結することで秘密情報 s を復元する．

2.2 暗号化と秘密分散法が可換な方式

文献 [8] ではクラウドサービスでの利用に適した秘密分散法について議論しており，既存の XOR 演算で構成される秘密分散法がクラウドサービスでの利用に適していることを示している．また，暗号化処理としてストリーム暗号を，秘密分散処理として XOR 演算で構成された秘密分散法を用いることで，暗号化（復号）処理と秘密分散による分散（復元）処理の順序を変更することが可能であることを示している．たとえば，式 (1) の s_i がストリーム暗号によって暗号化された情報であるとする．つまり，秘密情報 $s = s_1||s_2||\cdots||s_{n_p-1}$ と鍵 K から生成されたキーストリーム $k = k_1||k_2||\cdots||k_{n_p-1}$ の XOR 演算 $s_i = s_i \oplus k_i$ で表現できる．よって，部分分散情報 $w_{(i,j)}$ は以下のように表現できる（式 (2)）．

$$w_{(i,j)} = (s_{i-j} \oplus k_{i-j}) \oplus r_j^0 \oplus r_{i+j}^1 \quad (2)$$

また，XOR 演算の可換性より式 (3) が成立する．

$$((s_{i-j} \oplus k_{i-j}) \oplus r_j^0 \oplus r_{i+j}^1) \oplus k_{i-j} = s_{i-j} \oplus r_j^0 \oplus r_{i+j}^1 \quad (3)$$

式 (3) は暗号化された分散情報 $w_{(i,j)}$ に対して対応する部分キーストリーム k_i を XOR 演算している．つまり，分散情報に対してもストリーム暗号による復号が可能であることが分かる．したがって，暗号化→秘密分散と処理した場合でも，復号→復元の順で処理することが可能である．また，秘密分散→暗号化と処理した場合も同様に可能である．

表 3 $(3,n)$ 閾値秘密分散法の復元アルゴリズム
Table 3 Recovery algorithm of $(3,n)$-threshold scheme.

Input：$(w_{t_0}, w_{t_1}, w_{t_2})$
Output：s

$s_0 \in \{0\}^d$;
$x = t_1 - t_0$, where $t_0 < t_1$;
$y = t_2 - t_1$, where $t_1 < t_2$;

//**[Step 1]**
Divide w_{t_0} into $(w_{(t_0,0)}, \cdots, w_{(t_0,n_p-2)})$;
Divide w_{t_1} into $(w_{(t_1,0)}, \cdots, w_{(t_1,n_p-2)})$;
Divide w_{t_2} into $(w_{(t_2,0)}, \cdots, w_{(t_2,n_p-2)})$;

//**[Step 2]**
For$(i = 0; i < n_p - 1; i++)$\{
  $u_{(t_0,t_2,i)} = w_{(t_0,i)} \oplus w_{(t_2,i)}$;
  $u_{(t_1,t_2,i)} = w_{(t_1,i)} \oplus w_{(t_2,i)}$;
\}

//**[Step 3]**
$\alpha = 0$;
While$(\alpha(x+y) \not\equiv x \pmod{n_p})$ $\alpha++$;
For$(i = 0; i < n_p - 1; i++)$\{
  $v_{(t_0,t_2,i)} = 0$;
  If$(i + \beta(x+y) \not\equiv n_p - 1 \pmod{n_p}$, for $\beta = 0, \cdots, \alpha - 1)$\{
    For$(j = 0; j < \alpha; j++)$
      $v_{(t_0,t_2,i)} = v_{(t_0,t_2,i)} \oplus u_{(t_0,t_2,i+j(x+y))}$;
  \} Else \{
    For$(j = \alpha; j < n_p; j++)$
      $v_{(t_0,t_2,i)} = v_{(t_0,t_2,i)} \oplus u_{(t_0,t_2,i+j(x+y))}$;
  \}
\}

//**[Step 4]**
$\alpha = 0$;
While$(\alpha y \not\equiv -x \pmod{n_p})$ $\alpha++$;
For$(i = 0; i < n_p - 1; i++)$\{
  $v_{(t_1,t_2,i)} = 0$;
  If$(i + \beta y \not\equiv n_p - 1 \pmod{n_p}$, for $\beta = 0, \cdots, \alpha - 1)$\{
    For$(j = 0; j < \alpha; j++)$
      $v_{(t_1,t_2,i)} = v_{(t_1,t_2,i)} \oplus u_{(t_1,t_2,i+jy)}$;
  \} Else \{
    For$(j = \alpha; j < n_p; j++)$
      $v_{(t_1,t_2,i)} = v_{(t_1,t_2,i)} \oplus u_{(t_1,t_2,i+jy)}$;
  \}
\}

//**[Step 5]**
If$(t_0 + y \not\equiv n_p - 1 \pmod{n_p})$\{
  $j = t_0 + y$;
  $i = 0$;
  While$(j - 2ix \not\equiv n_p - 1 \pmod{n_p})$\{
    $s_{2(i+1)x} = v_{(t_0,t_2,j-2ix)} \oplus v_{(t_1,t2,j-2ix)} \oplus s_{2ix}$;
    $i++$;
  \}
\}
If$(t_0 + 2x + y \not\equiv n_p - 1 \pmod{n_p})$\{
  $j = t_0 + 2x + y$;
  $i = 0$;
  While$(j + 2ix \not\equiv n_p - 1 \pmod{n_p})$\{
    $s_{-2(i+1)x} = v_{(t_0,t_2,j+2ix)} \oplus v_{(t_1,t2,j+2ix)} \oplus s_{-2ix}$;
    $i++$;
  \}
\}

Return $s = s_1||\cdots||s_{n_p-1}$;

2.3 暗号化と秘密分散法を組み合わせた方式

暗号化と秘密分散法を組み合わせた研究として青野らの方式 [9] がある．この方式では秘密情報をバーナム暗号で暗号化したものに対して秘密分散法を実行することで分

散情報を生成する．秘密情報を復元する際は分散時に使用した鍵を用いて，秘密分散法→バーナム暗号による復号の順で復元する．これにより，仮に閾値分の分散情報を集められ復元されても，復元された秘密情報はバーナム暗号によって暗号化されているため，鍵が洩れない限り秘密情報が漏洩することはない．つまり，盗聴やなりすましに対する安全性を向上させることができる．また，(k, L, n) ランプ型閾値秘密分散法と組み合わせることにより，$k-t$ $(1 \leq t \leq L-1)$ 以上の分散情報から秘密情報が部分的に洩れてしまうという欠点を補いつつ，分散情報のデータ容量ををを n/L にすることができる．しかし，この方式ではバーナム暗号で使用した鍵の管理が必要となる．

3. 提案方式

3.1 概要

これまでの秘密分散法のみを利用したシステムモデル（以下，秘密分散方式と呼ぶ）では，クライアント端末（Client）上で秘密分散処理を行うことを想定している（**図 1**：破線）．それに対し本稿では，通信資源に制約があるモバイル環境でクライアント端末が利用されることを想定し，秘密分散処理を外部サーバ（Server）に委託することで，分散時のクライアント端末の転送負荷を軽減することを目的とする．このとき，近年のクラウドに対するセキュリティ問題に鑑み，委託先のサーバやストレージ（$\mathrm{Sto}_{1\ldots n}$）では管理者による覗き見の危険性があるという前提のもとで議論する．

こうした前提のもと，本稿では文献 [9] の暗号化と秘密分散法を組み合わせた方式を処理委託方式 [10] に適用する（以下，文献 [9] 改良方式と呼ぶ）ことで，委託先サーバ上での安全性を確保する（図 1：実線）．しかし，このままでは鍵管理コストが発生してしまう．そこで提案方式では鍵管理コストを削減するため，暗号化解除と呼ぶ処理を導入する．

本方式における暗号化解除処理とは，暗号化された秘密情報を秘密分散することによって得られる個々の分散情報を，（暗号化前の）秘密情報を秘密分散して得られる分散情報に復号することである．これにより，復元時は秘密分散法のみで秘密情報を復元できるため鍵管理の必要がない．暗号化解除の実現には 2.2 節で述べた XOR 演算の可換性を利用するため，本方式では暗号化方式としてストリーム暗号を，秘密分散法には XOR 演算のみで構成される方式を用いる．暗号化解除の詳細については 3.3 節で述べる．

本稿では説明のため，秘密情報を暗号化したデータを**暗号化情報**，秘密情報を秘密分散して得られるデータを**分散情報**，暗号化情報を秘密分散して得られるデータを**暗号化分散情報**と呼ぶことにする．

秘密分散法は機密性と可用性を改善する技術であり，データの改竄および誤りに対処する完全性については提供されない．本稿において，データの完全性は，電子署名や暗号学的ハッシュ関数 [13] などを用いる従来技術で対応することを想定している．なお，完全性に関する詳細な議論は本研究の焦点とは異なるため本稿では触れないこととする．

3.2 システム構成

Client，Frontend-Server（FS），Backend-Server（BS），Strorage（Sto）からなるシステムモデル（**図 2**）を想定し，処理内容と信頼性を次のように定義する．なお，それぞれの通信はすべて SSL で保護されているものとする．

- **Client：暗号化処理および復元処理**
  （処理内容）
  分散時は保存する秘密情報をストリーム暗号によって暗号化する．復元時は (k, n) 閾値秘密分散法によって秘密情報を復元する．
  （信頼性）
  Client はユーザが使用する端末を想定するため，ユーザにとって最も信頼性の高い安全なリソースである．
- **Frontend-Server：秘密分散処理**
  （処理内容）
  Client から受け取った暗号化情報に対し (k, n) 閾値秘密分散法を実行し，暗号化分散情報を生成する．

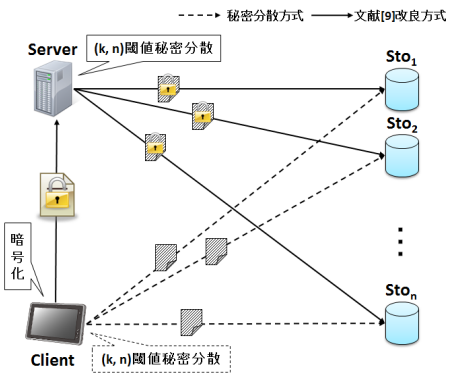

図 1 二方式のシステム構成例
Fig. 1 System configuration example of two scheme.

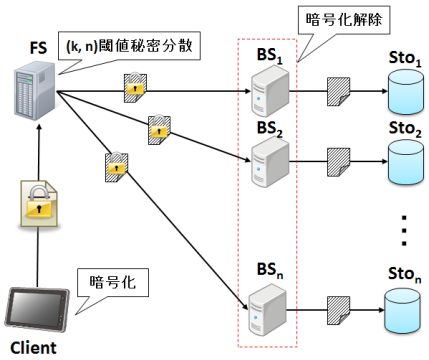

図 2 提案方式のシステム構成例
Fig. 2 System configuration example of our proposed scheme.

（信頼性）
Frontend-Server ではサーバ管理者による覗き見の危険性があるが，Backend-Server との結託はない．

- **Backend-Server：暗号化解除処理**
  （処理内容）
  暗号化分散情報を分散情報へと復号する．
  （信頼性）
  Backend-Server もサーバ管理者による覗き見の危険性があるが，Frontend-Server との結託はない．また，他の Backend-Server および Storage との結託により $k-1$ 個以上の分散情報を取得することはできない．
- **Storage：分散情報保管**
  （処理内容）
  分散情報を保管する．
  （信頼性）
  Storage もストレージ管理者による覗き見の危険性があるが，Backend-Server および他の Storage との結託により $k-1$ 個以上の分散情報を取得することはできない．

3.3 暗号化解除処理

XOR 演算の可換性から，ストリーム暗号と XOR 演算で構成された秘密分散法は処理の順序を変えることができる（2.2 節参照）．これを利用し，暗号化分散情報 x_i を分散情報 w_i と復号する処理を行うことで，暗号化の際に発生する鍵管理コストを削減する．

栗原らの $(3,n)$ 閾値秘密分散法 [4]（表 2）を例にあげると，式 (2) より，暗号化分散情報 x_i を構成する各部分情報 $x_{(i,j)}$ は次の式で表せる．

$$x_{(i,j)} = c_{i-j} \oplus r_j^0 \oplus r_{i+j}^1 \tag{4}$$
$$= (s_{i-j} \oplus k_{i-j}) \oplus r_j^0 \oplus r_{i+j}^1 \tag{5}$$

各 Backend-Server は Client で使用した同一の鍵 K から部分キーストリーム k_i を生成する．Frontend-Server から受け取った暗号化分散情報 x_i がどのアルゴリズムで生成されたデータであるか識別[*7]し，

$$x_{(i,j)} \oplus k_{i-j} = ((s_{i-j} \oplus k_{i-j}) \oplus r_j^0 \oplus r_{i+j}^1) \oplus k_{i-j}$$
$$= s_{i-j} \oplus r_j^0 \oplus r_{i+j}^1 = w_{(i,j)}$$

のように，対応する k_i を XOR することで部分分散情報 $w_{(i,j)}$ を復号できる．このとき，2.1 節より四則演算はすべて n_p を法としたものであることに注意する．

本稿では栗原らの $(3,n)$ 閾値秘密分散法 [4] を基に暗号化解除処理のアルゴリズムを作成しているが，XOR 演算のみで構成された秘密分散法ならばどの方式を用いても暗号化解除処理を導入することは可能である．ただし，キーストリーム k を分割するサイズや部分分散情報 $w_{(i,j)}$ の構成は，使用する秘密分散法のアルゴリズムやパラメータによって変わるため，暗号化解除処理のアルゴリズムはそれらに合わせて作成する必要がある．

具体例として栗原らの方式 [4] の $(3,4)$ 閾値秘密分散（$n=4$，$n_p=5$）を本方式で用いた場合を説明する．暗号化分散情報 x_0 は以下の 4 つの部分分散情報で構成されている．

$$x_{(0,0)} = c_0 \oplus r_0^0 \oplus r_0^1 \qquad (0-0 \bmod 5 = 0)$$
$$x_{(0,1)} = c_4 \oplus r_1^0 \oplus r_1^1 \qquad (0-1 \bmod 5 = 4)$$
$$x_{(0,2)} = c_3 \oplus r_2^0 \oplus r_2^1 \qquad (0-2 \bmod 5 = 3)$$
$$x_{(0,3)} = c_2 \oplus r_3^0 \oplus r_3^1 \qquad (0-3 \bmod 5 = 2)$$

x_0 を処理する Backend-Server は鍵 K によって生成したキーストリーム k を 4 分割[*8]して部分キーストリーム k_1〜k_4 を生成し，以下のように対応する k_i を $x_{(i,j)}$ に XOR 演算することで部分分散情報 $w_{(i,j)}$ を復号できる．

$$x_{(0,0)} = w_{(0,0)}$$
$$x_{(0,1)} \oplus k_4 = w_{(0,1)}$$
$$x_{(0,2)} \oplus k_3 = w_{(0,2)}$$
$$x_{(0,3)} \oplus k_2 = w_{(0,3)}$$

ただし，c_0 は [4] のアルゴリズム上，秘密情報 s とは独立に生成され，かつすべて 0 で構成された要素である．そのため $c_0 = s_0$ と扱うことができ，結果として $x_{(0,0)} = w_{(0,0)}$ となる．

3.4 分散過程

本節では提案方式の分散過程の詳細（**図 3**）について述べる．

(1) Client は暗号化に用いる鍵 K を保存ごとにランダムに生成する．また，分散情報の送信先となる n 個の

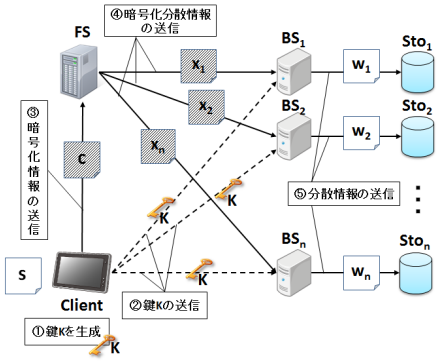

図 3 提案方式における分散処理
Fig. 3 Distribution process on our proposed scheme.

[*7] 各分散情報 w_i は構成される部分分散情報 $w_{(i,j)}$ が異なる（表 2 参照）．

[*8] 栗原らの方式における $(3,4)$ 閾値秘密分散法では秘密情報 s を 4 分割して部分秘密情報 s_1〜s_4 を得る．

表 4 実験環境
Table 4 Experiment environment.

	Client	Frontend-Server (FS)	Backend-Server (BS)
CPU	Pentium M 1.70 GHz	Intel(R) Core i7-3970 3.50 GHz	1vCPU
メモリ	1 GB	32 GB	2 GB
OS	CentOS 5.8	CentOS 6.4	CentOS 6.4
言語	Java（java-1.7.0-openjdk.x86_64）		

Backend-Server を決定する．

(2) Client は各 Backend-Server に対し認証要求を行い，同時に鍵 K を送信する．各 Backend-Server は Client を認証し，同時に鍵 K を受け取る．

(3) Client は秘密情報 s に対して，生成した鍵 K を用いてストリーム暗号で暗号化する．生成した暗号化情報 c は Frontend-Server へ送信し，その後，鍵 K を廃棄する．

(4) Frontend-Server は暗号化情報 c に対し (k, n) 閾値秘密分散法を実行し，暗号化分散情報 x_i を生成する．生成した暗号化分散情報 x_i はそれぞれ各 Backend-Server へ送信する．

(5) 各 Backend-Server は Client から受け取った鍵 K を用いて，Client のストリーム暗号で使用した同一のキーストリーム k を生成する．Frontend-Server から受け取った暗号化分散情報 x_i に対し，対応する k_i を XOR 演算することによって復号する．復号した分散情報 w_i は Storage へと送信し保管する．その後，鍵 K を廃棄する．

このように，提案方式では Client と各 Backend-Server で鍵 K を共有するため，通信を SSL で保護することで鍵 K の漏洩などの対策を行っている．

3.5 復元過程

復元過程は従来方式と同様に，Client で行う．

(1) Client は任意の k 個の分散情報 w_i を選択し，Storage へ認証要求する．

(2) Storage は Client を認証後，指定された分散情報 w_i を Client へ送信する．

(3) 取得した k 個の分散情報 w_i から (k, n) 閾値秘密分散法によって秘密情報 s を復元する．

4. 実験と考察

本章では提案方式を（通信部分を除く）処理時間と安全性の観点から考察する．さらに，秘密分散方式や文献 [9] 改良方式と比較することで提案方式の実用性を評価するとともに，提案方式で必要となる鍵の送信方法について考察する．

表 4 に図 2 における各構成要素の測定環境を示す．Backend-Server は VM として動作しており，VM の動作環境は，CPU：Xeon E5620 2.40 GHz，メモリ：8 GB，論理プロセッサ数：8 であり，ハイパーバイザとして VMware vSphere Hypervisor（ESXi）5.1.0 を使用している．

表 5 ファイルサイズに対する各処理時間
Table 5 Processing time for each file size.

ファイルサイズ	処理時間（秒）			
	暗号化 (1)	分散 (2)	暗号化解除 (3)	復元 (4)
1 KB	0.153	0.0002	0.158	0.0006
4 KB	0.154	0.0007	0.160	0.001
16 KB	0.164	0.003	0.177	0.007
64 KB	0.206	0.004	0.232	0.028
256 KB	0.313	0.006	0.283	0.117
1 MB	0.383	0.011	0.317	0.165
4 MB	0.680	0.035	0.392	0.435
16 MB	1.51	0.133	0.601	1.79
64 MB	5.33	0.560	1.53	8.61
256 MB	21.1	2.19	5.05	31.7
1 GB	83.9	8.26	20.6	123

4.1 提案方式の処理時間

(1) Clinet による暗号化処理，(2) FS による秘密分散法の分散処理，(3) BS による暗号化解除処理，(4) Client による秘密分散法の復元処理の個々の処理時間を表 4 の実験環境で測定した．**表 5** には 10 回試行した各処理の平均時間を示している．ただし，秘密分散法の分散処理で用いる真性乱数の生成時間はアルゴリズムによって時間差が大きいため測定時間には含めていない．本測定では，ストリーム暗号として AES の CTR モードを，秘密分散法として栗原らの $(3, 4)$ 閾値秘密分散法のアルゴリズム [4] を採用している．文献 [7] では実用的観点からパラメータとして $k = 3$，$n = 4$ が妥当であることが示されているため，本測定でもこのパラメータを採用した．

本方式で導入した暗号化解除処理は，暗号化処理に近い値となっている．これは暗号化解除におけるキーストリーム生成時間が XOR 演算の処理時間に比べて大きいためである．つまり，暗号化解除処理は使用するストリーム暗号の方式に依存する．

モバイル機器をビジネスで活用する場合，たとえば，外出先で記録したメモの保存（ファイルアップロード），また，資料確認や営業時のプレゼンテーション（ファイルダウンロード）などが利用シーンとして考えられる．このよ

表 6 各方式の評価
Table 6 Evaluation of each scheme.

	鍵管理コスト	Client 通信量	分散時の通信量	復元時の通信量	覗き見耐性	ストレージ間結託耐性
秘密分散方式	○	n	n	k	○	×
文献 [9] 方式	×	n	n	k	○	△
文献 [9] 改良方式	×	1	$n+1$	k	○ (△)*10	△
提案方式	○	1	$2n+1$ (or $n+1$)*9	k	○ (△)*10	×

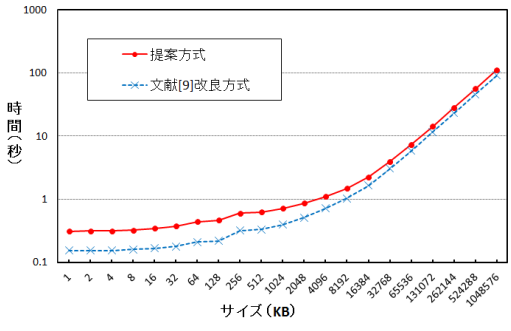

図 4 分散時における各方式の処理時間
Fig. 4 Processing time for each scheme on Distribution process.

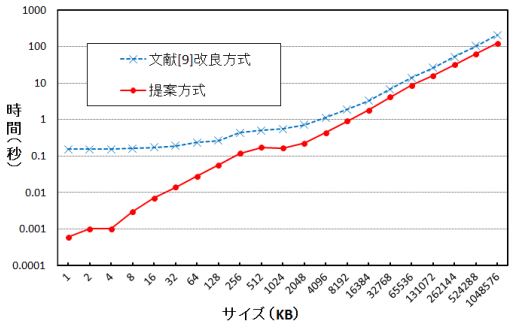

図 5 復元時における各方式の処理時間
Fig. 5 Processing time for each scheme on Recovery process.

うな利用シーンを想定し，ファイルサイズ 1 KB〜1 GB について表 5 に示した．

図 4 に文献 [9] 改良方式と提案方式の分散時における処理時間，**図 5** に復元時における処理時間を示す．これらは表 5 の (1)〜(4) の処理時間から算出した．提案方式の分散時の処理時間は，暗号化解除処理時間が加わるため，文献 [9] 改良方式と比較して約 22%大きくなる．一方，復元処理時間は約 40%小さくなる．提案方式は文献 [9] 改良方式に比べ，アップロード時の処理時間が大きくなるが，暗号化解除処理はクラウド上で実行されるためユーザエクスペリエンスに影響はない．

表 5 および図 4，図 5 における処理時間は，ファイルサイズに対して原則として線形に増加する．ファイルサイズが大きい区間（本実験環境では 16 MB 以上）においては，線形に増加していることが確認できる．一方，ファイルサイズが小さい区間（16 MB 以下）においては，処理の際に発生するオーバヘッドが処理時間に対して無視できなくなるため，非線形な増加を示すと考えられる．

4.2 提案方式の安全性

Frontend-Server，Backend-Server，Storage における提案方式の安全性を考察する．

- Frontend-Server
  Frontend-Server にわたる情報はストリーム暗号によって暗号化された暗号化情報 c である．そのため，Client 端末で使用した鍵 K が洩れない限り，秘密情報 s が漏洩する危険性はきわめて低い．
- Backend-Server および Storage
  各 Backend-Server にわたる情報は Client 端末で使用した鍵 K と，秘密分散法によって生成された暗号化分散情報 x_i である．Backend-Server は鍵 K を用いて暗号化分散情報 x_i を分散情報 w_i に復号はできるが，秘密分散法で生成される分散情報は情報理論的安全性を持つため，残り $k-1$ 個の分散情報が手に入らない限り，秘密情報 s は漏洩しない．Storage も w_i のみを得るため，同様に安全である．

ただし，Frontend-Server と Backend-Server で結託されると鍵 K と暗号化情報 c が得られるため，秘密情報 s が復元されてしまうという制限がある．

4.3 提案方式の評価

秘密分散方式，文献 [9] 方式，文献 [9] 改良方式，提案方式の比較を**表 6** に示す．秘密分散処理を委託する文献 [9] 改良方式や提案方式では，システム全体として見た通信量は間に処理サーバを挟んだ分，余分に多くなってしまう．しかし，クライアントから見た通信量は秘密情報と同サイズの暗号化情報を転送するのみであり，秘密分散方式や文献 [9] 方式に比べ Client 端末上の転送負荷を抑えることができる．さらに，一般的にクラウド上のサーバやストレージ間に確保される通信帯域は十分な大きさがあるため，余分に発生する通信量の影響は比較的小さいと考えられる．また，復元時の通信量は処理委託をすることで変わること

*9 提案方式は Backend-Server と Storage の機能を同一サーバ上に実装可能である．この場合，Backend-Server と Storage 間の通信がなくなるため，通信量は $n+1$ となる．
*10 処理委託方式は，秘密分散処理を行う委託先で計算量的安全性（△）を確保する．

はない．

一方，安全性の観点から比較すると，どの方式も情報理論的安全性（○）があるため，サーバ管理者による覗き見に対する安全性は確保される．しかし，ストレージ間の結託のような閾値分の分散情報が集められてしまう危険に対しては，文献 [9] 方式やその改良方式では仮に復元されたとしても暗号化による計算量的安全性（△）が確保されるためリスクは低いが，提案方式ではその耐性がない．その一方で，秘密分散方式や提案方式は鍵管理が不要なため，鍵を紛失して秘密情報を復元できなくなるというリスクがない．

このように，安全性の観点からいえば，文献 [9] 改良方式の方が処理委託方式としてはより情報漏洩のリスクは低い．しかし，表 6 からも分かるように，提案方式は秘密分散処理を委託するサーバに対して計算量的安全性を持ちつつ，ストレージに対して秘密分散方式と同等の安全性を実現している．

以上により，秘密分散方式が実用化されていることをふまえれば，提案方式をクラウドストレージサービスに適用することは十分可能である．さらに，クライアント端末の転送量の削減，鍵管理が不要であるという秘密分散法本来の利点から，提案方式が，通信帯域が乏しく紛失や盗難のリスクが高いモバイル機器での利用により適しているといえる．

4.4 鍵の送信方法

3.4 節でも述べているように，提案方式では秘密情報を保存するたびに Client は各 Backend-Server にストリーム暗号で用いる鍵を暗号化通信で送信しなければならない．そこで，複数の Backend-Server に対し効率的に鍵を送信する仕組みであるシングルサインオン（SSO）の利用を考えている．シングルサインオンは認証が必要な複数のサービスに対し一度の認証で利用可能とする技術である．また，秘密分散を利用した分散ファイルシステムに SSO の機能を加えたシステムも提案されている [10]．SSO の仕組みを用いれば，Client は認証サーバ（IdP）への一度の認証処理のみで，（SSO に対応した）複数の Backend-Server に対してストリーム暗号用の鍵を送信することが可能となると考えられる．

5. おわりに

本稿では暗号化と秘密分散法を組み合わせた方式を発展させ，鍵管理の不要な安全な処理委託方式を提案した．提案方式では秘密分散処理を外部サーバに委託することでクライアント端末上の転送負荷を削減するとともに，暗号化解除処理を導入することで暗号化時に発生する鍵管理コストをなくし，復元時の処理負荷の軽減を実現できる．さらに，提案方式を実用的観点から考察することで，秘密分散法のみのシステムモデルと同等の安全性を，委託先サーバに対して計算量的安全性を持つ処理委託方式で実現し，クラウドストレージサービスとしてモバイル機器での利用に適した方式であることを示した．

今後の課題として，認証処理や鍵配布，通信部分も含めた提案方式全体を評価することがあげられる．また，秘密分散法の復元処理の代理をサーバに委託することを考えた場合，同原理の方式によって委託処理に計算量的安全性を与える方式についても検討課題とする．

謝辞 本研究の一部は，日本学術振興会科学研究費補助金基盤研究（B）（課題番号 23300026，24300025）および若手研究（B）（課題番号 25730085）の助成を受けたものである．

参考文献

[1] Shamir, A.: How to share a secret, *Comm. ACM*, Vol.22, No.11, pp.612–613 (1979).
[2] 多田美奈子，藤井吉弘，保坂範和，栃窪孝也，加藤岳久：閾値 3 の秘密分散法の構成法，コンピュータセキュリティシンポジウム（CSS）2005 予稿集，Vol.2005, pp.637–642 (Oct. 2005).
[3] 藤井吉弘，栃窪孝也，保坂範和，多田美奈子，加藤岳久：排他的論理和を用いた (k, n) しきい値法の構成法，電子情報通信学会技術研究報告，Vol.107, No.44, pp.31–38 (2007).
[4] Kurihara, J., Kiyomoto, S., Fukushima, K. and Tanaka, T.: A Fast $(3, n)$-Threshold Secret Sharing Scheme Using Exclusive-OR Operations, *IEICE Trans. Fundamentals*, Vol.E91-A, No.1, pp.127–137 (2008).
[5] Kurihara, J., Kiyomoto, S., Fukushima, K. and Tanaka, T.: On a Fast (k, n)-Threshold Secret Sharing Scheme, *IEICE Trans. Fundamentals*, Vol.E91-A, No.9, pp.2365–2378 (2008).
[6] 山本博資：(k, L, n) しきい値秘密分散システム，電子通信学会論文誌，Vol.J68-A, No.9, pp.945–952 (1985).
[7] 松本 勉，清藤武暢，鴨志田昭輝，新谷敏文，佐藤 敦：セキュアデータ保管サービス向け高速秘密分散方式，第 29 回暗号と情報セキュリティシンポジウム SCIS2012 論文集，1E2-4 (2012).
[8] 須賀祐治：排他的論理和を用いた (k, n) 閾値秘密分散法の新しい構成とその優位性について，コンピュータセキュリティシンポジウム（CSS）2012 論文集，Vol.2012, pp.185–192 (Oct. 2012).
[9] 青野成俊，岩村恵市：実用観点からみた秘密分散法に関する一考察，コンピュータセキュリティシンポジウム 2009（CSS2009）論文集，Vol.2009, pp.601–606 (Oct. 2009).
[10] 熊谷悠平，西村浩二，大東俊博，近堂 徹，相原玲二：認証フェデレーションに基づく分散ファイル管理システムの提案，情報処理学会研究報告，Vol.2012-IOT-18, No.8, pp.1–6 (2012).
[11] 高荒 亮，岩村恵市：XOR を用いた高速な (k, L, n) ランプ型秘密分散法に関する研究，コンピュータセキュリティシンポジウム（CSS）2009 論文集，Vol.2009, pp.949–954 (Oct. 2009).
[12] 永井良英，高荒 亮，岩村恵市：XOR を用いた高速な秘密分散法のデータ容量削減に関する一手法，コンピュータセキュリティシンポジウム（CSS）2012 論文集，Vol.2012, pp.177–184 (Oct. 2012).
[13] Eastlake III, D. and Hansen, T.: US Secure Hash Algorithms (SHA and HMAC-SHA), RFC 4634 (July 2006).

吉田 耕太

2012 年広島大学工学部第二類（情報系）卒業．現在，同大学大学院総合科学研究科博士課程前期在学中．情報システム，情報セキュリティに関する研究に従事．

西村 浩二 （正会員）

1989 年広島大学工学部第二類（電気系）卒業．1991 年同大学大学院工学研究科博士課程前期修了．広島大学総合情報処理センター助手，同大学情報メディア教育研究センター准教授等を経て，2011 年より同教授．博士（工学）．コンピュータネットワークの運用管理，移動透過通信，情報セキュリティに関する研究に従事．電子情報通信学会会員．

大東 俊博 （正会員）

2002 年徳島大学工学部知能情報工学科卒業．2004 年同大学大学院工学研究科博士前期課程修了．2008 年神戸大学大学院自然科学研究科博士課程後期課程修了．現在，広島大学情報メディア教育研究センター助教．博士（工学）．暗号理論，ネットワークセキュリティ，認証プロトコルに関する研究に従事．電子情報通信学会 SCIS20 周年記念賞，SCIS2013 イノベーション論文賞を受賞．電子情報通信学会正員．

相原 玲二 （正会員）

1981 年広島大学工学部第二類（電気系）卒業．1986 年同大学大学院工学研究科博士課程後期修了．同大学助手，同大学集積化システム研究センター助教授を経て，現在，同大学情報メディア教育研究センター教授．工学博士．コンピュータネットワークに関する研究に従事．電子情報通信学会，IEEE Computer Society，IEEE Communications Society 各会員．