Vol.2013-IOT-22 No.15
2013/8/1

秘密分散法を利用したクラウドストレージサービスのための安全な処理委託方式

吉田 耕太[1]　西村 浩二[2]　大東 俊博[2]　相原 玲二[2]

概要：近年のモバイル端末の普及から，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可用性と秘匿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端末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，それに伴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，鍵管理の不要な安全な処理委託方式を提案し，その実用性を考察する．

キーワード：暗号化，秘密分散法，処理委託，鍵管理

Secure Outsourcing Scheme for Cloud Storage Services using Secret Sharing Scheme

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

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 which is unnecessary of key management by developing a method that combines encryption and SSS, and we consider its practicality.

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などのモバイル端末の急速的な普及により，モバイル端末をビジネス分野へ活用することに注目が集まっている．紛失や盗難による情報漏洩のリスクが高いモバイル端末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サーバへ秘密情報を保管することには，サーバ管理者の不正アクセス等による情報漏洩やデータ紛失などのセキュリティ面での不安や法律面での問題 [1] [*1] がある．秘密分散法は，

[1] 広島大学大学院総合科学研究科
Graduate School of Integrated Arts and Sciences, Hiroshima University
[2] 広島大学情報メディア教育研究センター
Information Media Center, Hiroshima University

[*1] 個人情報は暗号化等により秘匿化されているかどうかを問わず，サーバ管理者は個人情報と他の情報とを区別する必要があるため管理が難しい．

データの可用性・秘匿性を向上でき，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 [3] 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全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 [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 [*2] 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盛んに行われおり [7][8][9][10]，NRI セキュアが提供している SecureCube/SecretShare [*3] 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端末を想定しているが，モバイル端末はデスクトップ型 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す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，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．

[*2] 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．

[*3] http://www.nri-secure.co.jp/service/cube/secretshare.html

表 1　$(3,n)$ 閾値秘密分散法の分散情報の構成
Table 1　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}	$s_{n-1} \oplus r_0^0 \oplus r_{n-1}^1 \| \| s_{n-2} \oplus r_1^0 \oplus r_n^1 \| \| \cdots \| \| s_{n+1} \oplus r_{n_p-2}^0 \oplus r_{n-3}^1$

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栗原らの方式 [4][5] などが提案されている．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．

秘密情報 s を $s = s_1 \| s_2 \| \cdots \| s_{n_p-1}$ 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．d ビットの乱数 r_0^0〜$r_{n_p-2}^0$, r_0^1〜$r_{n_p-2}^1$ を独立に生成し，s_i, r_i^0, r_i^1 を XOR 演算することで部分分散情報 $w_{(i,j)}$ を生成する (式 1)．

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

各分散情報 w_i は表 1 のように部分分散情報 $w_{(i,j)}$ を連結させることで構成される．

但し，n_p は $n \leq n_p$ となる素数を，$\|$ はデータの連結を，$\oplus$ は XOR 演算を示し，s_0 は d ビット全て 0 で構成されているものとする．復元処理の詳細は紙面の都合上割愛する（文献 [4] 参照）．

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（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．例えば，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 $w_{(i,j)}$ は次のように表現できる（式 2)．

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

また，XOR 演算の可換性より式 3 が成立する．

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

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．従って，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．

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化したものに対して秘密分散法を実行する．これにより，仮に閾値分の分散情報を集められても，秘密情報は暗号化によって保護されるため，盗聴やなりすましに対する安全性を向上できる．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，部分的に情報が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，クライアント端末上で秘密分散処理を行うことを想定している（図 1：点線）．それに対し本稿では秘密分散処理を外部サーバに委託することで，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．この時，近年のクラウドに対するセキュリティ問題を鑑み，委託先のサーバではサーバ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．

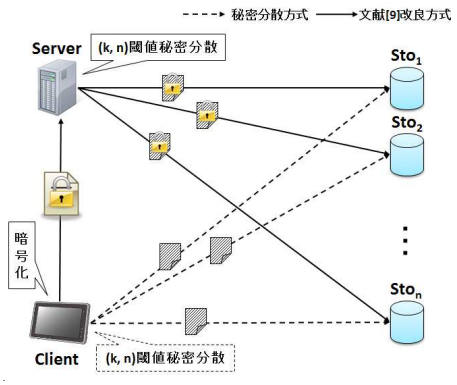

図 1 二方式のシステム構成例
Fig. 1 System configuration example of two scheme

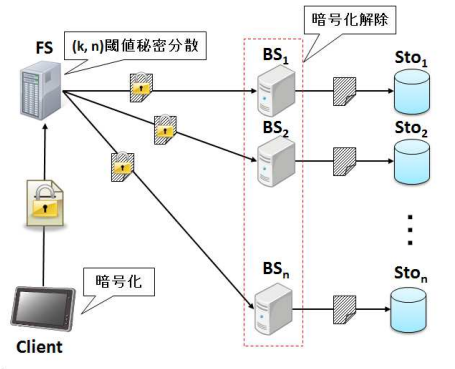

図 2 提案方式のシステム構成例
Fig. 2 System configuration example of our proposed scheme

3.2 システム構成

Client，Frontend-Server(FS)，Backend-Server(BS)，Strorage(Sto) から成るシステムモデル (図 2) を想定し，処理内容と信頼性を次のように定義する．

- **Client**：**暗号化処理および復元処理**
  分散時：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：(k, n) 閾値秘密分散法によって秘密情報を復元する．
  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とする．
- **Frontend-Server**：**秘密分散処理**
  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．
  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**
  暗号化分散情報を分散情報へと復号する．
  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
  分散情報を保管する．
  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 1）を例に挙げると，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．

$$\begin{aligned} x_{(i,j)} &= c_{i-j} \oplus r_j^0 \oplus r_{i+j}^1 \\ &= (s_{i-j} \oplus k_{i-j}) \oplus r_j^0 \oplus r_{i+j}^1 \end{aligned}$$

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 [*4] し，

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

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．ただし，k を分割するサイズや $w_{(i,j)}$ の構成は使用する秘密分散法のアルゴリズムや k, n などの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) を本方式で用いた場合，暗号化分散情報 x_0 は，

$$\begin{aligned} x_{(0,0)} &= c_0 \oplus r_0^0 \oplus r_0^1 \\ x_{(0,1)} &= c_4 \oplus r_1^0 \oplus r_1^1 \\ x_{(0,2)} &= c_3 \oplus r_2^0 \oplus r_2^1 \\ x_{(0,3)} &= c_2 \oplus r_3^0 \oplus r_3^1 \end{aligned}$$

の部分分散情報で構成されている．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 [*5] 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．

[*4] 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 1 参照)．
[*5] 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．

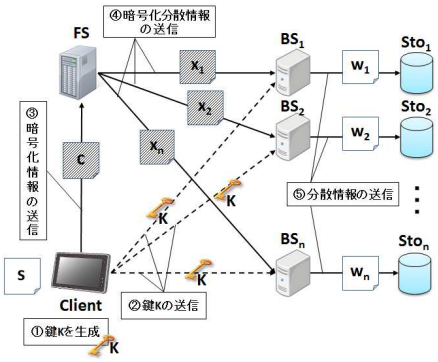

図 3 提案方式における分散処理
Fig. 3 Distribution process on our proposed scheme

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

この時，s_0 はアルゴリズム上全て 0 で埋められているため，$c_0 = s_0$ となり，処理する必要はない．

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 を保存毎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 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．

3.5 復元過程

復元過程は従来方式と同様に，Client にて行う．

(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．

4.1 提案方式の処理時間

(1) 暗号化処理，(2)（秘密分散法による）分散処理，(3) 暗号化解除処理，(4)（秘密分散法による）復元処理の個々の処理時間を測定した．表 2 には次の実験環境で 10 回試行した各処理の平均時間を示している．但し，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．また，暗号化方式には AES の CTR モードによるストリーム暗号を，秘密分散法には栗原らの $(3, 4)$ 閾値秘密分散法のアルゴリズム [4] を採用している．

-VMS
CPU：Intel(R) Xeon(R) CPU E5620 2.40GHz,
RAM：8GB，論理プロセッサ：8
Hypervisor：VMware vSphere Hypervisor (ESXi) 5.0.0
-VM
CPU：1vCPU，RAM： 2GB，OS： CentOS 6.4,
言語：Java（java-1.7.0-openjdk.x86_64）

表 2 ファイルサイズに対する各処理時間
Table 2 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)
1KB	0.159	0.0002	0.158	0.0006
1MB	0.325	0.029	0.312	0.037
1GB	21.0	14.5	20.4	13.8

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．

また，図 4 に文献 [9] 改良方式と提案方式の分散時における全体の処理時間を，図 5 に復元時における全体の処理時間を示す．これらは (1)〜(4) の個々の処理時間の和で算出している．分散時において，提案方式は暗号化解除処理がある分大きくなってしまうが，4MB までは 1 秒とかからず，文献 [9] 改良方式との差は 256MB でも約 5 秒程度である（図 4）．よって，暗号化解除による影響は十分許容範囲

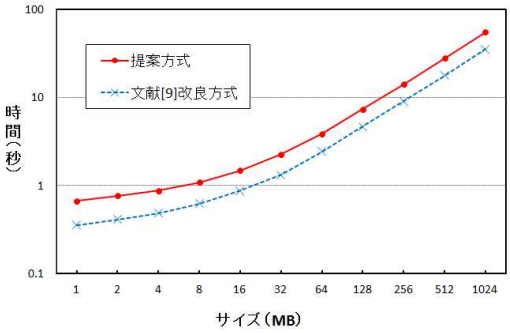

図 4 分散時における各方式の処理時間
Fig. 4 Processing time for each scheme on Distribution process

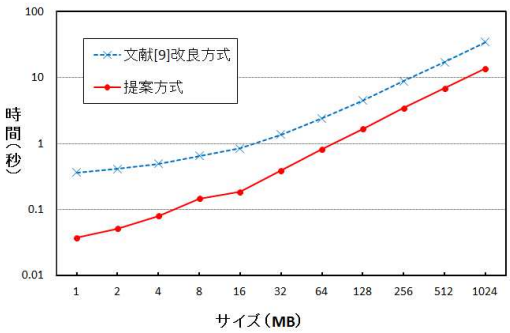

図 5 復元時における各方式の処理時間
Fig. 5 Processing time for each scheme on Recovery process

であると言える．一方，復元時において，提案方式は復号処理が無い分，文献 [9] 改良方式よりも高速である（図 5）．

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 に渡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極めて低い．
- Backend-Server および Storage
  各 Backend-Server に渡る情報は 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．

但し，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．

表 3　各方式の評価
Table 3　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	○（△）*7	△
提案方式	○	1	$2n+1$（or $n+1$）*6	k	○（△）*7	×

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 3 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だけなので，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，安全性の観点から言えば，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 3 からも分かるように，提案方式は少なくとも秘密分散方式と同等の安全性を確保し，秘密分散方式が実用化されていることを踏まえれば，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端末での利用により適していると言える．

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共に，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無くし，復元時の処理負荷の軽減を実現した．また，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があり，クラウドストレージサービスとしてモバイル端末での利用に適した方式であることを示した．

今後は通信部分も含めた，システム全体を通しての提案方式の具体的実装方法やその評価，本稿では特に取り上げなかった復元時への応用が課題となる．

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) 及び若手研究 (B)(課題番号 25730085) の助成を受けたものである．

*6　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．
*7　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．

参考文献

[1] 経済産業省個人情報保護ガイドライン. http://www.meti.go.jp/policy/it_policy/privacy/kojin_gadelane.htm.
[2] 秘密分散に関する技術ガイドラインおよび秘密分散技術利用に関するガイドライン. http://www.jipdec.or.jp/archives/ecom/results/h21seika/H21results-10.pdf.
[3] A. Shamir. How to share a secret. *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613, 1979.
[4] J. Kurihara, S. Kiyomoto, K. Fukushima, and T. Tanaka. A (3, n)-threshold secret sharing scheme using exclusive-or operations. *IEICE Trans. on Fundamentals*, Vol. E91-A, No. 1, pp. 127–137, 2008.
[5] J. Kurihara, S. Kiyomoto, K. Fukushima, and T. Tanaka. On a fast (k, n)-threshold secret sharing scheme. *IEICE Trans. on Fundamentals*, Vol. 91-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. *SCIS2012*, Vol. 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. *CSS2012*, pp. 185 – 192, October 2012.
[9] 青野成俊, 岩村惠市. 実用観点からみた秘密分散法に関する一考察. *CSS2009*, Vol. C6-2, , October 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.