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の実装と評価

吉田 耕太† 西村 浩二‡ 大東 俊博‡ 相原 玲二‡

†広島大学大学院総合科学研究科 739-8521 広島県東広島市鏡山 1-7-1
‡広島大学情報メディア教育研究センター 739-8511 広島県東広島市鏡山 1-4-2

あらまし 秘密分散法を利用したクラウドストレージサービスが注目されている．秘密分散法はデータの可用性と秘匿性を向上できる一方，データ容量が増えてしまうために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．それに対し我々は既に，秘密分散処理をクラウド上の代理サーバに委託する方式（処理委託方式）によって端末の通信量を抑える方式を提案している．提案方式は XOR で構成された秘密分散法とストリーム暗号を用いることで代理サーバに対して計算量的安全性を確保すると共に，秘密鍵の管理を不要としている．しかし，認証処理やファイル転送時間も含めたサービス全体としての処理時間の評価が行われていなかった．そこで本稿では，提案方式のプロトタイプシステムを実装し，その性能について評価する．

Implementation and Evaluation of Secure Outsourcing Scheme for Cloud Storage Services using Secret Sharing Scheme

Kouta Yoshida† Kouji Nishimura‡ Toshihiro Ohigashi‡ Reiji Aibara‡

†Graduate School of Integrated Arts and Sciences, Hiroshima University
1-7-1 Kagamiyama, Higashihiroshima, Hiroshima 739-8521, JAPAN
‡Information Media Center, Hiroshima University
1-4-2 Kagamiyama, Higashihiroshima, Hiroshima 739-8511, JAPAN

Abstract Cloud storage service using Secret Sharing Scheme (SSS) attracts attention. SSS can improve the availability and confidentiality of data, but the scheme is not suitable for processing by mobile devices of poor communication bandwidth due to increased data capacity. In contrast, We have already proposed the outsourcing scheme that outsource SSS to agent server on cloud to reduce traffic of device. The proposal scheme ensures computationally secure for agent server by using stream cipher and SSS using exclusive-OR operations, and is unnecessary of private key management. However, evaluation of the processing time as the whole service including attestation processing or file transfer time was not performed. In this paper, we implement the prototype system of proposal scheme and evaluate its performance.

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ビジネス活用に注目が集まっている．モバイル端末はネットワークに接続できる環境であれば場所・時間を問わず様々なサービスを受けることができ，その中でも業務データなどの重要な秘密情報をバックアップやファイル共有などの用途で，安全かつ手軽に利用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，サーバ障害による

データ紛失や管理者の覗き見による情報漏洩などのセキュリティ面での不安や，法律面での問題 [1] がある．秘密分散法は，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 倍に増えるという問題がある．これらの問題に対して，高速な秘密分散処理を実現する排他的論理和（XOR）のみで構成された秘密分散法 [4][5] や，分散情報のデータ容量を n/L 倍に抑える (k, L, n) ランプ型閾値秘密分散法 [6] が提案されている．さらに近年では，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ており [7][8][9][10]，NRI セキュアが提供している SecureCube/SecretShare[11] 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．

秘密分散法を利用したシステムモデルの多くはユーザが使用するクライアント端末で秘密分散処理を行う．しかし，デスクトップ型 PC やノート PC に比べて通信帯域に余裕がないモバイル端末での処理を想定した場合，ランプ型といえどデータ容量が増えてしまう秘密分散法をモバイル端末上で処理するのは効率的ではない．加えて，移動が伴うモバイル端末では，常にネットワークに接続できるとは限らないため，なるべく早くモバイル端末上での処理を終えたいというニーズもある．

そこで著者らは暗号化と秘密分散法を組み合わせた，秘密鍵の管理が不要な安全な処理委託方式を提案している [12]．この方式では秘密分散処理を委託することによる通信量の削減や，暗号化による委託先の計算量的安全性を確保している．さらにストリーム暗号と XOR で構成された秘密分散法の可換性を利用した暗号化解除処理を導入することで，秘密鍵の管理を不要としている．しかし，文献 [12] では認証処理やファイル転送を含めた全体の処理時間の評価は行っていなかった．そこで本稿では，文献 [12] の方式を拡張した提案方式のプロトタイプシステムを実装し，その性能について評価する．

表 1: $(3, n)$ 閾値秘密分散法の分散情報の構成

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

2 既存研究

2.1 XOR で構成された (k, n) 閾値秘密分散法

排他的論理和 (XOR) で構成された (k, n) 閾値秘密分散法については栗原らの方式 [4][5] や松本らの方式 [7]，須賀の方式 [8] などが提案されている．これらの方式は処理の高速な XOR のみで秘密分散法を構成することで，Shamir の方式の計算負荷の問題を解消している．具体例として栗原らの $(3, n)$ 閾値秘密分散法 [4] における分散情報の構成例を表 1 に示す．このとき，s_i，r_j^i はそれぞれ秘密情報，乱数を $n_p - 1$ で分割した部分情報を，w_i は分散情報を示し，n_p は $n \leq n_p$ となる素数を，$||$ はデータの連結を，$\oplus$ は XOR 演算を示す．但し，s_0 は全て 0 で構成されているものとする．

2.2 暗号化と秘密分散法が可換な方式

文献 [8] では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 で構成された秘密分散法を利用すれば，暗号化（復号）処理と秘密分散による分散（復元）処理の順序の変更が可能であることを示している．つまり，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，すなわち，秘密分散法で生成した分散情報に対して暗号化方式による復号処理が可能となる．

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 n/L にすることができる．しかし，この方式では暗号化で使用した秘密鍵の管理が必要となる．

3 提案システム

3.1 目的

クライアント端末で秘密分散処理を行う従来の秘密分散法のシステムモデル（図 1 破線：以下，秘密分散方式と呼ぶ）に対し，本研究では秘密分散処理をサーバに委託する [10] 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．

さらに，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に発展させる（以下，文献 [9] 改良方式と呼ぶ）ことで，委託先サーバに対して計算量的安全性を確保する（図 1：実線）．

3.2 提案方式の概要

文献 [9] 改良方式では秘密鍵の管理が必要となる．これは，データのやり取りが頻繁で扱う秘密鍵が多い場合，その管理コストが大きくなってしまう．これに対し文献 [12] では鍵管理の不要な処理委託方式を提案している．

提案方式 [12] では秘密鍵の管理を不要とするため，暗号化解除と呼ぶ処理を導入している．暗号化解除とは，暗号化→秘密分散を経て得られる分散情報に対し暗号化方式による復号処理を行うことである．復号には暗号化で使用したものと同一の鍵を用いることで行う．これにより，復元時は秘密分散法のみで秘密情報を復元

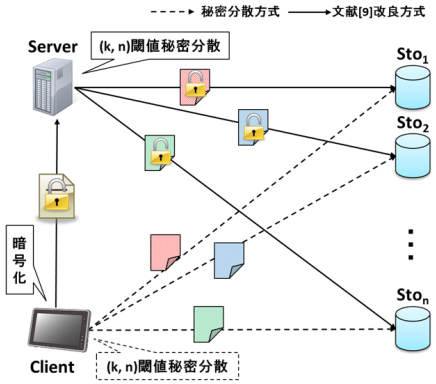

図 1: 二方式のシステム構成例

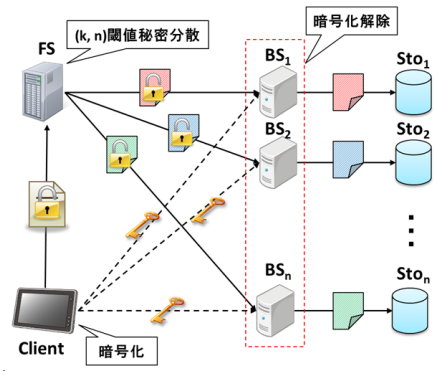

図 2: 提案方式のシステム構成例

できるため鍵管理の必要がない．提案方式 [12] では暗号化解除実現のため，2.2 節で述べた暗号化処理と秘密分散処理が可換な，ストリーム暗号と XOR で構成された秘密分散法を用いる．暗号化解除の詳細については 3.4 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**，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**，暗号化分散情報を復号したデータを**分散情報**と呼ぶことにする．

3.3 提案方式の設計

3.3.1 システム構成

Client，Frontend-Server(FS)，Backend-Server(BS)，Storage(Sto) から成るシステムモデル (図 2) を想定し，それぞれの機能と信頼性を次のように定義する．なお，それぞれの通信は全て SSL で保護されている．

- **Client**：**暗号化処理および復元処理**

  (機能：分散時) 保存したい秘密情報をストリーム暗号で暗号化する．ストリーム暗号に用いる秘密鍵や保存セッション毎にユニークなセッション ID（SID）はランダムに生成される．ここで，SID は 32 バイトなど十分に長いデータとする．

  まず，分散情報の保存先となる n 個の Storage の URI（StoURI_x, $x = 1, 2, ..., n$）およびそこに代理で保存する Backend-Server の URI（BsURI_x, $x = 1, 2, ..., n$）を決定する．さらに，各 Backend-Server 単位のセッション ID（SID_x, $x = 1, 2, ..., n$）を SID を鍵，BsURI_x をメッセージとして HMAC-SHA256 から得られたハッシュ値から生成する．ここで生成された SID_x は Frontend-Server と各 Backend-Server 間の認証に用いられる．

  Client は各 Backend-Server に対してユーザ認証をした上で StoURI_x と SID_x，秘密鍵を登録し，その後，Frontend-Server に対してユーザ認証をした上で暗号化情報と StoURI_x，n 個の BsURI_x を送信する．

  (機能：復元時) 任意の k 個の Storage からユーザ認証をした上で分散情報を受信し，(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**

  (機能) Client から受信し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．各 Backend-Server には生成した各暗号化分散情報と，Client と同様に SID と BsURI_x から計算した $\mathrm{SID}x$ を送信する．

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

- **Backend-Server**：**暗号化解除処理**

  (機能) Backend-Server は Frontend-Server から受信した SID_x が Client が登録したものと一致することを確認する．一致したならば，Frontend-Server から受信した暗号化分散情報を Client から受信した秘密鍵を用いて分散情報へと復号する．その後，Backend-Server は Storage に分散情報を保存する．

  (信頼性) Backend-Server も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**

  (機能) Backend-Server から受信した分散情報を保管する．

  (信頼性) Storage も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．

3.3.2 分散情報管理

本稿ではクラウド上の複数のストレージにデータを保管することを想定しているため，分散配置されたデータを管理するシステムが必要である．そこで提案方式では，熊谷らが提案している分散ファイル管理システム [10] を利用する．このシステムは，秘密分散法で生成された分散情報を管理することを想定しており，分散 KVS（キーバリューストア）で分散配置された情報を管理する．また，各 Storage が連携して KVS を運用できるようにすることでスケールアウトが容易なシステムを実現している．

3.4 暗号化解除処理

XOR の可換性から，ストリーム暗号と XOR 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 [8]．これを利用し，分散時に暗号化分散情報を分散情報へ復号する処理を入れることで，暗号化の際に発生する鍵管理を不要とする．

栗原らの $(3, n)$ 閾値秘密分散法 [4] を例に説明する．この方式では，暗号化分散情報 x_i を構成する各部分情報 $x_{(i,j)}$ は以下の式で表現できる（式 (1)）．

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

この時，s_i，k_i はそれぞれ秘密情報とストリーム暗号で XOR されたキーストリームの部分情報を示す．

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 を生成し，$x_{(i,j)}$ と k_i を XOR する（式 (2)）．

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

このように，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，$w_{(i,j)}$ を連結することで分散情報 w_i を得る．

但し，$w_{(i,j)}$ 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] を $n = 4$ で用いた場合を説明する．暗号化分散情報 x_0 は，

$$\begin{aligned} x_{(0,0)} &= s_0 \oplus r_0^0 \oplus r_0^1 \\ x_{(0,1)} &= (s_4 \oplus k_4) \oplus r_1^0 \oplus r_1^1 \\ x_{(0,2)} &= (s_3 \oplus k_3) \oplus r_2^0 \oplus r_2^1 \\ x_{(0,3)} &= (s_2 \oplus k_2) \oplus r_3^0 \oplus r_3^1 \end{aligned}$$

の 4 つの部分情報で構成されている [*1]．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 [*2] 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 することで部分分散情報 $w_{(i,j)}$ を復号できる．

$$\begin{aligned} x_{(0,1)} \oplus k_4 &= s_4 \oplus r_1^0 \oplus r_1^1 = w_{(0,1)} \\ x_{(0,2)} \oplus k_3 &= s_3 \oplus r_2^0 \oplus r_2^1 = w_{(0,2)} \\ x_{(0,3)} \oplus k_2 &= s_2 \oplus r_3^0 \oplus r_3^1 = w_{(0,3)} \end{aligned}$$

3.5 秘密分散過程の処理手順

本節では提案方式における秘密分散過程の処理手順 (図 3) について述べる．但し，図 3 は 1 台の Backend-Server との処理手順を示しているが，実際には n 台の Backend-Server と並行して処理する．

[*1] アルゴリズム上，s_0 は独立に生成されるため部分キーストリーム k_i は XOR されていない．

[*2] 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．

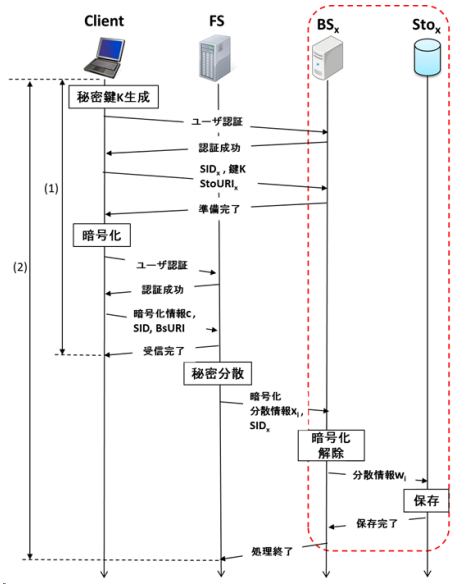

図 3: 提案方式における秘密分散過程の処理手順

1. Client はセッション ID（SID）と暗号化に用いる秘密鍵 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 の URI（BsURI$_x$）を決定し，各 Backend-Server に対してユーザ認証を行う．

2. 各 Backend-Server への認証成功後，Client は保存先となる Storage の URI（StoURI$_x$）を決定し，StoURI$_x$ と秘密鍵 K，ハッシュ値 SID$_x$ を各 Backend-Server へ送信する．各 Backend-Server は受信したこれらの情報を保持しておく．

3. Client は秘密情報 s に対して秘密鍵 K を用いてストリーム暗号で暗号化する．その後，Frontend-Server に対してユーザ認証を行い，認証成功後，生成した暗号化情報 c と SID，n 個の BsURI$_x$ を Frontend-Server へ送信し，秘密鍵 K を廃棄する．

4. Frontend-Server は受信した暗号化情報 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．その後，生成した暗号化分散情報 x_i とハッシュ値 SID$_x$ をそれぞれ各 Backend-Server へ送信する．

5. 各 Backend-Server は Frotned-Server から受信した SID$_x$ が Client が登録したものと

表 2: 実験環境

	Client	Frontend-Server	Backend-Server
CPU	Pentium M 1.70GHz	Intel Core i7-3970 3.50GHz	1vCPU
メモリ	1GB	32GB	2GB
OS	CentOS 5.8	CentOS 6.4	CentOS 6.4
通信速度	100Mbps		

一致するか確認し，一致すれば暗号化分散情報 x_i を受信する．その後，保持していた秘密鍵 K を用いて x_i に対し暗号化解除処理を実行し，分散情報 w_i を生成する．その後，w_i を Storage へ送信し，SID_x や秘密鍵 K を廃棄する．

3.6 復元過程の処理手順

提案方式における復元過程の処理手順について述べる．なお，復元過程は従来の秘密分散方式と同様の手順となる．

1. Client は任意の k 個の分散情報 w_i を選択し，保存先の各 Storage に対しユーザ認証を行う．
2. 認証成功後，各 Storage は Client へ分散情報 w_i を送信する．
3. Client は取得した 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．

3.7 実装

Client，Frontend-Server，およびBackend-Server は Java（java-1.7.0-openjdk.x86_64）で実装し，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, n) 閾値秘密分散法 [4] のアルゴリズムを採用した．また，各端末間の通信は HTTPS で実装している．なお，Backend-Server は 4 台の VM 上にそれぞれ実装し，Backend-Server と Storage は同一端末上に実装している．

4 評価

本章では実装した提案方式を評価するため，(1) Client 上の処理時間と (2) 保存までに要する全体の処理時間を表 2 の実験環境の基で各ファイルサイズごとに測定し，秘密分散方式との比較を行った．加えて，安全性の観点からも秘密分散方式や文献 [9] 改良方式などと比較した．

なお，Backend-Server を実装した 4 台の VM は 2 台の VMS 上でそれぞれ 2 台ずつ稼働させている．VMS の環境は CPU：Xeon E5620 2.40GHz，メモリ：8GB，論理プロセッサ数：8 で，ハイパーバイザとして VMware vSphere Hypervisor (ESXi) 5.1.0 を使用している．

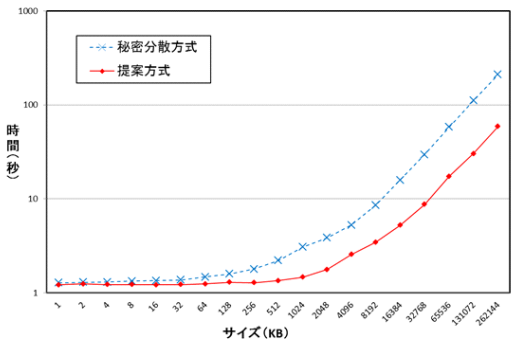

図 4: Client 上の処理時間

4.1 Client の処理時間

秘密分散方式と提案方式の Client 上で要する処理時間（図 3 の (1) の区間）を測定し，その結果を図 4 に示す．各測定値は 10 回試行した平均値で，秘密分散法は $n = 4$ で実行している．

図 4 より，提案方式が秘密分散方式に比べて処理負荷がが抑えられていることが確認できる．特に，ファイルサイズが大きくなるほど通信時間が支配的になるため，転送量の少ない提案方式の方がより速く Client 上での処理を終えることができる．

4.2 全体の処理時間

提案方式の保存までにかかる全体の処理時間（図 3 の (2) の区間）を測定し，その結果を図 5

表 3: 各方式の評価

	秘密分散方式	文献 [9] 方式	文献 [9] 改良方式	提案方式
鍵管理コスト	○	×	×	○
Client 通信量	n	n	1	1
分散時の通信量	n	n	$n+1$	$n+1$ (or $n+1$ [*3])
復元時の通信量	k	k	k	k
覗き見耐性	◎	◎	◎ (△) [*4]	◎ (△) [*4]
k 個以上の分散情報が漏洩した場合の耐性	×	△	△	×

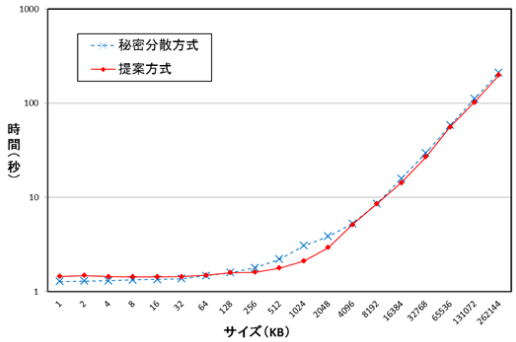

図 5: 全体の処理時間

に示す．測定方法は 4.1 節と同様で，秘密分散方式の値は 4.1 節のものと同じである．

図 5 より，秘密分散方式に比べ処理量・通信量共に多い提案方式は，秘密分散方式とほぼ同じ値となっている．これは本実験で用いた Client の端末よりも Frotned-Server の端末の方がより高性能で，秘密分散処理や転送処理がより高速に処理できたためである．

4.3 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評価する．

- 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情報である．そのため，Client で使用した秘密鍵が洩れない限り，秘密情報が漏洩する危険性は極めて低い．

- Backend-Server および Storage
  各 Backend-Server に渡る情報は秘密鍵と，秘密分散法によって生成された暗号化分散情報である．Backend-Server は秘密鍵を用いて暗号化分散情報を分散情報に復号はできるが，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，秘密情報は漏洩しない．Storage も分散情報のみを得るため，同様に安全である．

但し，Frontend-Server と Backend-Server で結託されると秘密鍵と暗号化情報が得られるため，秘密情報が復元されてしまうという制限がある．

4.4 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．

まず通信量の観点から評価する．処理委託方式である文献 [9] 改良方式や提案方式の通信量は，秘密分散方式や文献 [9] 方式に比べて多くなってしまう．しかし，クラウド上のサーバは一般的にユーザが使用するクライアント端末よりも高性能であり，4.2 節の結果から処理委託方式も従来方式と同じもしくはより高速に処理ができると予想できる．さらに，これら処理委託方式では 4.1 節の結果から，秘密分散方式に比べより速くクライアント端末上での処理を終えることができる．また，復元時の通信量はどの方式も同じである．

一方，安全性の観点から評価すると，どの方式もストレージ上において秘密分散法による情報理論的安全性（◎）を確保するため，管理者による覗き見に対する安全性は確保される．し

[*3] 提案方式における Backend-Server と Storage の機能を同一端末上に実装した場合の通信量．

[*4] 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．

かし，万が一分散情報が漏洩し，閾値分の分散情報が集められ復元された場合，文献 [9] 方式やその改良方式では復元された情報は暗号化による計算量的安全性（△）が確保されるため漏洩するリスクは低い．しかし，秘密分散方式や提案方式では復元された時点で秘密情報の漏洩に繋がるため，その耐性がない．その反面，文献 [9] 方式やその改良方式では秘密鍵を管理する必要があるが，秘密分散方式や提案方式ではその管理が不要であるという利点がある．

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安全である．しかし，表 3 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．従って，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と考える．さらに，クライアント端末上の処理負荷の削減，秘密鍵の管理が不要であるという利点から，提案方式が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

本稿では文献 [12] の方式を拡張した提案方式のプロトタイプシステムを実装し，その性能評価を行った．処理時間の測定の結果，提案方式ではユーザが使用するクライアント端末上の処理時間は従来の方式よりも高速に処理できることが確認できた．また，全体を通した処理時間においても，委託先に高性能なサーバを利用することで従来方式とほとんど変わらない処理時間であった．従って，提案方式は十分実用化可能な方式であると共に，移動の伴うモバイル端末での処理により適した方式であることが確認できた．

今後の課題として，本稿でできなかった分散ファイル管理システム [10] の実装や，複数のクライアント端末で同時に使用した場合の動作検証と性能評価を行うことが挙げられる．また，秘密分散法の復元処理を同原理の方式によって代理サーバに委託した場合や，認証機能として学認の SSO 機能を追加したシステムの評価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) 及び若手研究 (B)(課題番号 25730085) の助成を受けたものである．

参考文献

[1] 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.

[11] セキュアクラウドストレージサービス. http://www.nri-secure.co.jp/service/cube/secretshare.html.

[12] 吉田耕太, 西村浩二, 大東俊博, 相原玲二. 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. 情報処理学会研究報告, Vol. 2013-IOT-22, No. 15, pp. 1–6, 2013.