複数プロバイダに亘る分散ストレージのためのグループ横断秘密分散法

柯, 陳毓弢
九州大学システム情報科学府

穴田, 啓晃
九州先端科学技術研究所

川本, 淳平
九州先端科学技術研究所

Kirill, Morozov
Institute of Mathmatics for Industry, Kyushu University

他

出版情報：暗号と情報セキュリティシンポジウム. 2016, pp.3A1-3-, 2016-01-19. IEICE Technical Committee on Information Security (ISEC)
バージョン：
権利関係：

SCIS 2016 2016 Symposium on Cryptography and Information Security
Kumamoto, Japan, Jan. 19- 22, 2016
The Institute of Electronics, Information and Communication Engineers

複数プロバイダに亘る分散ストレージのためのグループ横断秘密分散法
Cross-group Secret Sharing for Distributed Storages over Providers

柯 陳毓弢* 穴田 啓晃† 川本 淳平† モロゾフ キリル‡ 櫻井幸一†
Chenyutao Ke* Hiroaki Anada† Junpei Kawamoto† Kirill Morozov‡ Kouichi Sakurai†

あらまし データをクラウド上のストレージに保存する場合，データの漏洩および損失は主要な安全性の課題である．本研究では，データの所有者がデータを異なるプロバイダのストレージに分散させ保存する設定において，データを漏洩と損失から保護する，グループ横断秘密分散法を提案する．従来の秘密分散法では，シェアの数が膨大になると，契約すべきプロバイダの数が膨大になる問題があり，もしシェアの数を抑えるとデータの損害率も上がる問題がある．この問題に対し，グループ横断秘密分散法では，データの所有者はシェアを少数のプロバイダに保存でき，なおかつ一つのプロバイダに閾値以上のシェアを預けないで済む．この特徴を有するグループ横断秘密分散法の構成には，Shamir の秘密分散法が二つ組み合わされる．データに対する秘密分散に加え，権限秘密に対する秘密分散が導入され，権限秘密のシェアが異なるプロバイダに配分される．

キーワード クラウドストレージサービス，データの漏洩，データの損失，秘密分散

1 はじめに

近年，高性能サーバおよび超高速ネットワークの発展に伴い，商用クラウドサービスが増加している．クラウドサービスが出現する以前では，ユーザは自分のプライベートな情報をローカル端末や他のデータ記憶装置に保存していた．ところが，現在，サービスプロバイダが提供するクラウドストレージを利用することが普遍的になってきた．

一方で，クラウドサービスプロバイダという第三者を利用することで新たに発生する課題もある．まず，利用するクラウドサービスが半永久的に利用できるという保証はなく，ある日サービスが終了してしまうかもしれない．また，適切なバックアップを保証するサービスばかりではなく，事故や災害などによりクラウドサービスに預けているデータが失われてしまう可能性がある．そのため，データ損失に対する対策が必要である．次に，情報セキュリティの観点から見れば，データの漏洩と改ざんなどのようなリスクを解消する必要がある．その結果，クラウドサービスにおける情報損失及び情報漏洩対策は，重要なトピックの一つとなっている．

情報損失の対策としては，一つのデータに対して複数のコピーを作成し異なる地域に配置されたサーバに保存する方法が考えられる．事故や災害などにより，データが損失を受けたとしてもコピーのうち一つが残っている限り，データの復旧が可能である．しかしながら，この方法では盗まれる可能性のあるデータ数が増えているため，情報漏洩のリスクを高めてしまう．

そこで我々は，安全なデータ保存技術は，同時に次の 2 つの性質を満たす必要があると考える．

1) 一つのデータをいくつかの分散情報に分割でき，一部の分散情報が破壊されても，データが利用可能である．
2) 一つのデータをいくつかの分散情報に分割でき，一部の分散情報が盗まれても，データが漏洩しない．

この 2 つの性質を満たす秘密分散法という技術が存在しており，Shamir 秘密分散法[1]その代表である．Shamir 秘密分散法では，一つの秘密情報を n 個の異なるシェアと呼ばれる分散情報に分割する．このうち，少なくとも k 個のシェアを集めれば秘密を復元できる．また，$k-1$ 個までシェアが漏洩しても，一切情報が漏洩しないことが情報理論的に保障されている手法である．なお，n と k はあらかじめ定めるパラメータであり，特に k は閾値と呼ばれる．

この秘密分散法に利用した商用クラウドデータスト

* 九州大学システム情報科学研究院，福岡県福岡市西区元岡 744 番地，
† 九州先端科学技術研究所，福岡市早良区百道浜 2 丁目 1 番地 22 号 福岡 SRP センタービル 7 階
‡ 九州大学マス・フォア・インダストリ研究所福岡県福岡市西区元岡 744 番地

レージサービスはいくつか提供されている [2]．ここうした商用ストレージサービスの多くでは，適切な閾値を設定するステップまでを考慮し，シェアの分配先に関する考慮は不十分である．前述のように，クラウドサービスには，サービス終了問題やサービスプロバイダにおける内部犯による情報漏洩という問題があり，その解決が必要だからである．

このように，完全には信用できないクラウドサービスを利用し，上記の要件１），２）を満たすより安全なデータストレージを用意するために，我々は複数のプロバイダにまたがったサービスが必要であると考える．本研究で提案する安全なクラウドストレージサービスでは，事故や災害などのリスクを減らすため地域的に分散させ，またプロバイダのリスクを減らすため，複数のプロバイダに分散させシェアを配置する秘密分散手法である．

地理的及び複数プロバイダに分散してシェアを配置する最も簡単な方法は，図１に示すように一つの地域に一つプロバイダを利用し１つのシェアを分配することである．しかし，このような単純な配置は，地域やプロバイダに対するリスク評価が異なる場合，最適な配置ではない．例として，今後３０年の間に災害の発生する確率が地域 a では 10%，地域 b では 30%，地域 c では 60%である３地域と，同じく今後３０年間の何らかの問題が起きうるリスク確率の評価値がそれぞれ 60%，30%，10%のプロバイダ x, y, z に対してシェアを分配する問題を考える．図１のように，各地域に一つずつプロバイダを用意しシェア配分する場合，シェアの分配数は３であり，閾値として２を設定したとする．この時，今後３０年の間に元の情報が復元できなくなる確率 P_1 は P_1 = 64.4608%である．一方，図２のように，リスクを考慮し分散することを考える．この場合，総シェア数は６であるから，閾値として４を選んだとする．この時，元の情報が復元できなくなる確率は P_2= 53.0668%となる．このように，シェアの配置にはリスクを考慮して行う必要がある．

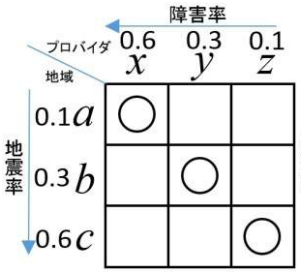

図 1：(2-out-of-3)

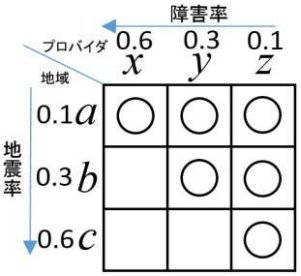

図２：(4-out-of-6)

もし，単純に Shamir の秘密分散法を用いて作ったシェアで一つのプロバイダに複数のシェアを配置すれば，２つの問題が起きるかもしれない．

１）一つのプロバイダに閾値個のシェアが配置されたら，このプロバイダが自分だけで秘密を復元できる．

２）プロバイダに複数のシェアが配置されたら，プロバイダの間で結託して秘密を復元できるリスクが上がる．

例えば，６個のシェアを図２のように配置されたら，従来の方法では，z から情報が漏えいすればシェアは３個流出したと考えていたが，提案手法を使えば同じ３個のシェアが流出したけど，秘密を復元されたリスクからみれば，１個シェアしか流出していないことになる．つまり，提案手法を使ったら，同じの数のシェアが流出した場合，秘密を復元されたリスクは，従来方法より遥かに低いということである．

2　背景と動機

本論文では，秘密情報（以降単に「秘密」と呼ぶ）を Shamir の(k, n)閾値秘密分散法を用いて n 個のシェアに分割する．シェアをクラウドプロバイダへ分配する際には次の２点に注意を払うべきである．

１）n 個のシェアを m 個の異なるプロバイダにを分配すること．

もっと簡単に言えば，シェアが盗まれることを考慮すると，シェアにプロバイダを分配させて保存すること．つまり，n 個のシェアを一つのプロバイダに分配するより m 個の異なるプロバイダにを分配するほうが情報漏洩のリスクが低いことは明らかになる．

2) n 個のシェアを l 箇所異なる領域にを分配すること．もっと簡単に言えば，シェアが失われる可能性を考慮すると，シェアを地理的に分配させて保存すること．2015 天津爆発, 2011 年福島第一原子力発電所事故の被害資料に参照によって，近隣のサーバに格納されたデータすべて破壊された．したがって，データ回復を確実にするために，我々は l 異なる領域にシェアを分配することを考慮した．

ここで，異なる領域にある異なるプロバイダにシェアを保存する一つのナイーブ方法がある．すなわち，$n = m = l$．しかし，現実的にはプロバイダの数が限られているので，シェアの数はプロバイダの数よりも大きくなる可能性がある．つまり，$m < l$，$l \leq n$ のことになる．そこで，本稿では，$m < n$ の状況を検討した．

2.1　既存研究

Smith らが階層秘密分散法（Layered-SSS）を提案した [3]．階層秘密分散法は，第１レベルのシェアを第２レベルの秘密として階層的にシェアを生成することを可能にしており，生成されたシェアを分散的に異なる領域に格納することが提案されている．

Zhang [4] と Martin は[5], シェアの数が増加によるスキームが破る危険性に対して，閾値を上げる方法を提案した．彼らの方法はシェアの増分で，(t, n) 閾値スキームから（t', n'）閾値スキームへの変更する方式である．しかし，閾値を上げるための計算量が大きいという欠点が存在している．

表 1. 自分の提案と既存提案の比較

名前	Layered SS [18]	Linear SS [16]	Threshold-Changeable SS [19,20]	CGSSS
直接にシェアを生成	×	○	○	○
直接にシェアを復元	×	○	○	○
シェア更新	×	×	○	○
閾値更新	-	-	○	×

Beimel は，一般的なアクセス構造を持つ線形秘密分散法（Linear-SSS）を提案した[6]．しかし，彼の提案は，シェアの増加に伴い，秘密にアクセスのパターンが極端に大きくなる欠点がある．

2.2 我々の提案と既存研究の比較

我々の提案では，データ漏洩とデータ損失に対して，ディーラはクライアントと見られ，2 つの基本機能である「ディーラ機能」と「リカバラ機能」を設計した．

ディーラ機能は，シェアの生成と分配する役目を務める．既存研究では，1 つのプロバイダのシェアの数の増加に伴に，閾値が固定なので，スキームの安全性が低下になる．スキームの安全性を維持するために，動的に閾値を変更すべきだと考えられる．一方，我々の提案は，スキームの安全性はシェアの数に依存しない．つまり，プロバイダのシェアの数はどのような変化しても，スキームの安全性が影響されない．我々の提案は，(t, n)閾値を(t, m)-(k, n)結束された閾値で置換する．((t, m)-(k, n)は，2 つの閾値スキームを連合されたという意味である) 我々のスキームは Zhang らの提案に比べ，増加したシェアの数を気にする必要はない．また，我々の提案のディーラ機能が低計算に実行できる．Beimel らの Linear-SSS は，同じの問題を解決できるが，シェアの数が大きくなると，秘密にアクセス構造のパターンが極端に増えるので，計算速度が極端に遅くなる．

リカバラ機能とは，シェアから秘密を復元することである．誰かが秘密を復元したいとき，彼は k 個以上異なるプロバイダからの k 個以上権限シェアと t 個以上データシェアを取得する必要がある． 2 つの条件が満足されていない場合は，誰も他の方法でデータ秘密を復元することができない. この点は, 別の提案には, 存在しない.

良いオンラインデータストレージサービスは，新シェアを追加する能力[7][8], 漏洩されたシェアを破棄する能力，およびシェアのなりすましを検出能力を備えるべきである[9] [10] [11]．そこで，我々の提案には，すべて持っている．

表 1 には，我々の提案と別の提案の比較を示す.

3 提案の設定及び攻撃パターン

3.1 モデルと仮定

本稿では， m 個のグループすなわちプロバイダと n 台サーバ及び一つのディーラからなるクラウドストレージサービスを考える．言い換えれば，我々は，ディーラが一つの秘密に n 個シェアを作り，シェアを分配する役目を与え，m 種類のプロバイダから合計 n 台サーバを借りて，一つのクラウドデータストレージサービスを構築する．

グループに P_i を表し，集合 $A = \{P_1, P_2, ..., P_m\}$ を m 個のグループからなる．また，我々はセキュリティパラメータ k を（k, m）閾値秘密分散法の閾値を表示し，秘密 v を共有する．ここで，我々は v を権限秘密という名前を付ける．

我々は，各グループにいくつかのデータストレージサーバを持っていることを想定し，m 個のグループに合計 n 台のサーバを持つとする. ここで, サーバの集合 $B = \{Q_1, Q_2, ..., Q_n\}$ を表す．また，我々はセキュリティパラメータ t を（t, n）閾値秘密分散法の閾値を表示し，秘密 s を共有している．ここで，我々は s をデータ秘密という名前を付ける．

我々の目的は，データ秘密 s を復元したい人は，必ず k 個のグループの承諾を持っている上で，t 個以上のデータシェアでデータ秘密 s を復元する．なぜ我々はそれを行う理由は，以下の 2 つの攻撃パターンから考える．

1） 不正のプロバイダ攻撃

元の Shamir 秘密分散法では，一つのプロバイダに閾値個シェアを預ければ，そのプロバイダが秘密を復元できる能力を持つことになる．もし，そのプロバイダが不正になったら, ユーザのデータが漏洩されるかもしれない. そのような危険を避けるために，我々は一つのプロバイダのみデータ秘密を復元することを禁止する．

2） 第三者攻撃

独断何か意図を持つ個人がプロバイダのサーバに侵入してデータを盗むことである．しかし，我々はこの攻撃者の攻撃能力は有限と想定する．すなわち，この攻撃者は, k 個のプロバイダの合計 $m-1$ 台までのサーバに侵入できるが，m 台以上のサーバに侵入できない．

図 3 は，同じグループ内のサーバは同じ権限秘密を持っている上で，異なるシェアを保持している．例としては，プロバイダ 1 のすべてのサーバはの権限シェア v_1 を持っており，異なるデータシェア p_i を持っている．

集合 A の中のサーバはすべて集合 B に所属し, 各プロバイダが任意に独自のサーバにアクセスできるが，他のプロバイダのサーバにアクセスすることはできない．あるプロバイダのサーバが置き換えられるとき，古いサーバ上のすべてのデータが簡単に新しいサーバに移動され

ると仮定する．プロバイダは新しいサーバを追加したい場合，我々は2つのケースを考えた．

1）プロバイダにすでに閾値個以上シェアを持っている場合，グループは簡単にShamir秘密分散法で新しいシェアを作成する．

2）グループのシェアの数が閾値未満の場合，他のグループと協力して新シェアを作成する．

ここで，一つの注意点がある．後者の場合には，グループのシェアが暗号化されなかったら，プロバイダが新シェアを作成できる上で，Shamir秘密分散法で秘密を復元できる．これがユーザとして望ましくない．

3.2 暗号化ツール

我々の提案はシャミールの秘密分散法に基づいている．この論文を通して，p と q は2つの大きい素数と設定し，q は $p-1$ を完全に割り切れる．G_q は，次数 q の Z^*_p のユニークなサブグループであり，g が G_q の生成元である．要素 $a \in Z_p$ が G_q に所属するかどうかは以下の式で簡単にテストできる．

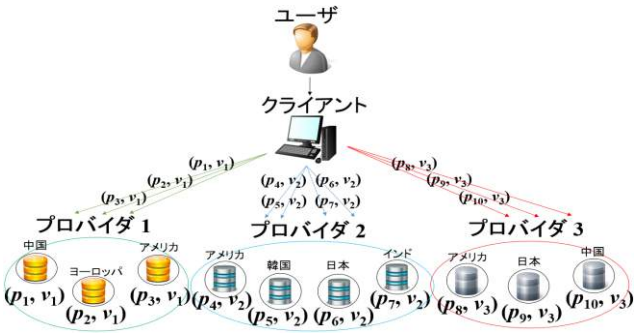

図3．提案データストレージサービスのサンプル

$$a \in G_q \leftrightarrow a^q = 1$$

すべて G_q から生成された b に対して($b \neq 1$)，a の離散対数はベース b で生成され($a \in G_q$)，$log_b(a)$で表される．

x は任意の整数だったら，$|x|$ が x のバイナリ表現の長さで示されている．

4 計算量上安全性を持つCG-SSS

本稿では，我々は(t, m)-(k, n)秘密分散法(CG-SSS)を提案した．この二つの閾値はすべてShamir秘密分散法に基づいていた．二つの閾値が同時に満たされた際のみ，データ秘密が正しく復元される．

このセクションでは，我々は(t, m)-(k, n)秘密分散法の計算量上安全性を実現する一つの方法を説明する．また，攻撃者の計算能力は多項式時間で問題を解決できることを前提とする．

4.1 定義

このセクションでは，我々はディーラをクライアントと見られ，シェアを生成し，分配し，データ秘密 s を復元する機能とする．我々は秘密鍵を持つ一方向関数を利用する．この秘密鍵を持つ一方関数は $f_h(x)$と表示され，h は秘密鍵と見られる．データ秘密 s に n 個のシェアを分け，i 番目のシェアを s_i と示す．権限秘密 v に m 個のシェアを分け，i 番目のシェアを v_i と示している．

我々はデータ秘密と権限秘密2つの秘密を設計する理由は，真のデータ秘密を隠すためである．真のデータ秘密を隠すために，我々は仮のデータ秘密 p を作る．

$$p = s - q$$

上式の q は，秘密鍵を持つ一方関数に権限秘密 v を代入して計算された中間値である．仮のデータ秘密 p に n 個シェアを分け，i 番目のシェアを p_i と示す．実際に，各プロバイダに分配されたデータシェアと権限シェアは，仮のデータシェア p_i と権限シェア v_i である．

4.2 計算量の安全性

誰かがデータ秘密を復元したいとき，必ず閾値個のグループに参加しなければならない前提を設定し，権限秘密 v はデータ秘密 s にアクセスできる前提条件とされる．我々は権限秘密 v を用いて権限シェア v_i を作成し，各プロバイダに分配する．しかし，各プロバイダが権限シェアを保管する間で，権限シェア v_i が攻撃者に盗まれる危険性が存在している．攻撃者が十分な仮のデータシェアと権限シェアを集めれば，データ秘密を復元することが可能になる．それを防ぐために，我々は秘密鍵 h を持つ一方向関数に権限秘密 v で中間値 q を計算し，その中間値 q でデータ秘密を隠す．このようにするメリットは，たとえ仮のデータシェアと権限シェアが漏洩されても，秘密鍵 h が攻撃者に知らない場合，攻撃者はデータ秘密を復元できない．

次は，計算上安全性を持つCGSSSの2つフェーズを説明する．

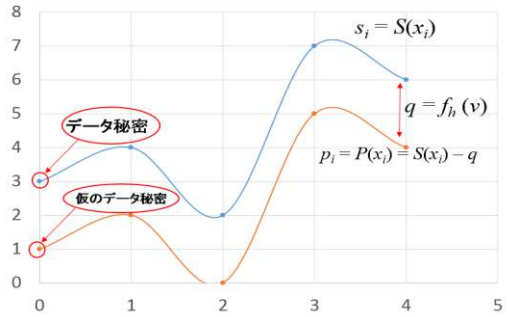

図4．秘密鍵ありラグランジュ多項式のイラスト

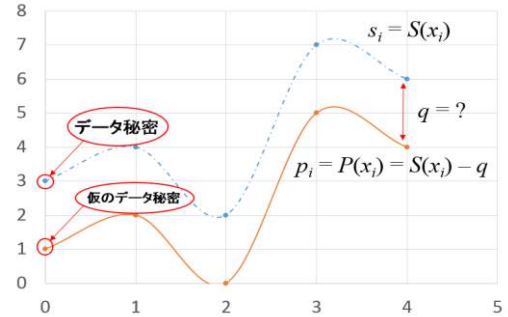

図5．秘密鍵なしのラグランジュ多項式のイラスト

「分散フェーズ」

1. ディーラはランダムで秘密鍵 h を生成する．秘密鍵の長さは適切に設定することができる．
2. 中間値 q は，以下の式に従って計算する．
$$q = f_h(v)$$
ここで，$f_h(x)$は秘密鍵 h を持つ一方関数と指す．
3. 仮のデータ秘密 p を下式のように計算する．
$$p = s - q$$
4. ディーラは Shamir 秘密分散法を用いて，仮のデータシェア p_i と権限シェア v_j を作る．
5. 最後, ディーラは各グループにp_iとv_jを分配する．

「復元フェーズ」

1. リカバラは，各グループから仮のデータシェア p_i と権限シェア v_j を取得する．
2. リカバラは，仮のデータシェア p_i を用いて，仮のデータ秘密 p を復元する．
(シェア p_i の数が閾値 k 以上ではないといけない．)
3. リカバラは，権限シェア v_i を用いて，権限秘密 v を復元する．
(シェア v_i の数が閾値 t 以上ではないといけない．)
4. リカバラは，一方関数 $f_h(x)$に秘密鍵 h で代入して，中間値 q を計算する．
$$q = f_h(v)$$
5. 最後，リカバラは式 $s = p + q$ のようにデータ秘密 s を復元する．

4.3 実際アプリケーションで提案の分析

図４では，仮のデータシェア p_i とデータシェア s_i の関係を示す．我々の提案では，最も重要なことは，秘密鍵 h であり，最も高いセキュリティレベルを有することである．私たちが知っているように，オンラインストレージ製品を利用する際に，ユーザが事前に登録したアカウントでログインしてサービスをする流れになる．図４の示すようにデータ秘密（赤い丸でマークされたところ）を復元するために，関数 $S(x_i)$もしくは $P(x_i)$でデータ秘密を復元できる．関数 $S(x_i)$と関数 $P(x_i)$を作るために，ラグランジュ補間が使用される．我々の提案では，ラグランジュ補間を二回使用して，仮のデータ秘密を求める関数 $P(x_i)$と権限秘密を求める関数 $V(x_i)$を再構築した．関数 $P(x_i)$と $V(x_i)$を求めてから，仮のデータ秘密 p と権限秘密 v を復元できる．本来なら，仮のデータ秘密と権限秘密があれば，データ秘密を復元するスキームを提案したいが，データの安全性の考慮した上で，元のデータ秘密を復元するための流れにユーザがコントロールできる要素である秘密鍵 h を加えた．誰かがデータ秘密を復元したい場合，必ず秘密鍵 h の取得が必要である．もし，秘密鍵 h を持っていなかったら，中間値 q を計算できずデータ秘密を $s = p + q$ ように復元できない．具体的に復元できないことは，図５のように示されている．

図４では，データ秘密 $s = S(0)$ が $q = f_h(v)$を用いてマスクされた．我々は直接に $p = s - v$ 式のように秘密をマスクされなかった理由は，権限秘密が異なるデータの秘密を隠すために再利用される可能性があり，権限シェアが敵に漏れるかもしれない．従って，我々は秘密鍵を持つ一方向関数を使用しないと，攻撃者が容易にデータ秘密 s を復元できる．

5 情報理論上安全性を持つ CG-SSS

セクション３では，我々は CG-SSS の計算上の安全性を説明した．しかし，近年，PC やスーパーコンピュータの発展により，計算上の安全性が不十分であることと指摘された．特に，一方関数を選択する時，適切な関数を選ばないとスキームが簡単に破られるかもしれない．特に，SHA1 一方関数の不安全性はすでにセキュリティの専門家に警告された．ここで，我々は無限の計算能力を持つ敵に対して，情報理論上安全性を持つ CG-SSS を提案した．

情報論理を持つ安全性

我々はワンタイムパッドという技術を利用してグループ横断横断秘密分散法（CGSSS）に情報理論的安全を保障できる．しかし，ワンタイムパッドを使用することの欠点がある．この欠点は，秘密鍵長と暗号化される平文の長さが同じサイズである．これによって，我々のスキームは他の正常な暗号化技術よりもはるかに秘密鍵のストレージスペースを保存する可能性がある．その問題に対して，我々は直接にデータ秘密にワンタイムパッドで暗号化しなく，権限秘密に暗号化して，権限秘密に暗号化された結果でデータ秘密を暗号化する．これによって，秘密鍵のサイズが一定範囲で抑える．

次は，情報理論性上安全性を持つ CG-SSS の２つフェーズを説明する．

「分配フェーズ」

1. ディーラは，権限秘密 v と同じ長さの乱数 r を生成する．
2. ディーラは，中間値 q を以下の式に従って計算する．
$$q = v \oplus r$$
(⊕はビット単位で排他的論理和演算である．)
3. ディーラは，仮のデータ秘密 p を式 $p = s - q$ に従って計算する．
4. ディーラは, Shamir 秘密分散法で仮のデータシェア p_i と権限シェア v_j を作る．
5. ディーラは，各プロバイダに仮のデータシェア p_i と権限シェア v_j を分配する．

誰かが秘密 s を復元したいなら，彼はまず中間値 q を計算すべきである．中間値 q を計算するために，彼は秘密鍵 r にディーラに秘密鍵の渡しの依頼が必要である．

「復元フェーズ」

1. リカバラは，仮のデータシェア p_i と権限シェア v_i

を異なるプロバイダから取得する.

2. リカバラは，仮のデータシェア p_i を用いて，仮のデータ p を復元する.
   (シェア p_i の数が閾値 k 以上ではないといけない．)
3. リカバラは，権限シェア v_j を用いて権限秘密 v を復元する.
   (シェア v_i の数が閾値 t 以上ではないといけない．)
4. リカバラは，式 $q = v \oplus r$ のように権限秘密 v を用いて中間値 q を計算する.
5. 最後に，リカバラは，式 $s = p + q$ のようにデータ秘密を復元する.

6 CG-SSS の安全性の分析

グループ横断秘密分散法では，正しくデータ秘密を復元するために，2 つの閾値(t, m)-(k, n)を同時に満たされなければならない.

我々の提案の安全性は，以下 3 つのパラメータに基づいた.

1. ディーラが持つ秘密鍵.（計算安全性では，秘密鍵 h と表示され，情報論理安全性では，秘密鍵 r と表示される）
2. 仮のデータシェア.
3. 権限シェア

定義 $m \geq 2$，m はグループの総数と指し，$t \leq m$，t は権限秘密の閾値と仮定する． $n \geq m$，n はサーバの総数と指し，1 つのグループに対して少なくとも一台サーバあり，k はデータ秘密の閾値である. 権限シェアの集合は，$VV = \{v_1, v_2, ..., v_m\}$とする. 仮のデータシェアの集合は，$PP = \{p_1, p_2, ..., p_n\}$とする.

例: 我々は図 3 のモデルの示すようにパラメータに $n = 10$，$m = 3$，$k = 7$，$t = 2$ を設定する.

状況 I 攻撃者は秘密鍵を取得できるが，権限シェアと仮のデータシェアを取得できない.

上の状況から分析すれば，攻撃者は明らかにデータ秘密を復元できない. 何故かと言うと，式 $s = p + q$ により，p と q の両方が攻撃者に知られていない時，データ秘密 s を復元することができない.

状況 II 攻撃者は閾値個以上の権限シェアを取得できるが，秘密鍵と閾値個以上の仮のデータシェアを取得できない.

例えば，攻撃者は権限シェア v_1 と v_2 を取得したら，ラグランジェ補間で権限秘密 v を復元できる.

$$VV = \{v_1, v_2\} \rightarrow v$$

しかし，攻撃者は秘密鍵を持っていないゆえに，権限秘密から中間値 q を計算することがきないので，式 $s = p + q$ から s を計算できない.

$$v \nrightarrow q$$

その結果，データ秘密を復元できない.

状況 III 攻撃者は閾値個以上の仮のデータシェアを取得できるが，秘密鍵と閾値個以上の権限シェアを取得できない.

例えば，攻撃者は仮のデータシェア $p_1, p_2, p_3, p_4, p_5, p_6, p_7$ を取得できるが，閾値個の権限シェアを取得できない. 攻撃者は，ラグランジェ補間で仮のデータ秘密を復元できる.

$$PP = \{ p_1, p_2, p_3, p_4, p_5, p_6, p_7 \} \rightarrow p$$

権限シェアの数が閾値未満なので，権限秘密 v を復元できない.

$$VV = \{ v_1, v_2, \ldots, v_i \} \nrightarrow v \{ i \leq 1 \}$$

攻撃者は権限秘密 v を復元できないので，中間値 q を計算できないので，式 $s = p + q$ からデータ秘密 s を復元することが不可能である.

状況 IV 攻撃者は，閾値個以上の仮のデータシェアと閾値個以上の権限シェアを取得できるが，秘密鍵を取得できない.

例えば，攻撃者は，仮のデータシェア p_1，p_2，p_3，p_4，p_5，p_6，p_7 と権限シェア v_1,v_2 を取得した．攻撃者は，ラグランジェ補間で仮のデータ秘密と権限秘密を復元する.

$$PP = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\} \rightarrow p$$
$$VV = \{v_1, v_2\} \rightarrow v$$

攻撃者が秘密鍵を持っていないので，権限秘密 v から中間値 q を計算できないので，式 $s = p + q$ に従ってデータ秘密 s を復元できない.

状況 5 攻撃者は，秘密鍵，閾値個以上の仮のデータシェアと閾値個以上の権限シェアをすべて取得できる.

例えば，攻撃者は，p_1，p_2，p_3，p_4，p_5，p_6，p_7 のデータシェアと権限シェア v_1，v_2 を取得した.

攻撃者は，ラグランジェ補間で仮のデータ秘密 p と権限秘密 v を復元する.

$$PP = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\} \rightarrow p$$
$$VV = \{v_1, v_2\} \rightarrow v$$

攻撃者は，秘密鍵を持っているので，中間値 q を計算でき，式 $s = p + q$ からデータ秘密 s を復元できる.

以上状況らから見ると，状況 5 のみ，攻撃者はデータ秘密を復元できる．攻撃者が我々のスキームを破りたい場合，彼は三つのパラメータを得らないといけない．第一，攻撃者は，秘密鍵を取得するために，秘密鍵の保存先に侵入して鍵を盗むもしくは秘密鍵が転送された時に，通信路を盗聴して秘密鍵を盗む．通常では，ディーラが攻撃者に支配されたら，すべて防御措置が終わるので，本稿では，ディーラが攻撃者に支配された状況を討論しない．攻撃者が通信路で秘密鍵を盗聴しないように，秘密鍵を第三者に渡す時に暗号化が必要される．第二，攻撃者はデータ秘密 p を取得するために，グループのサーバに侵入して t 個以上の仮のデータシェアを取得すべきである．第三，攻撃者は権限秘密 v を取得するために，グループのサーバに侵入して k 個以上の権限シェアを取得すべきである．攻撃者は上記の 3 つの条件を同時に満たすことが極める困難なので，我々は提案した方式が高い安全性を持つことを言える.

7　実験的評価

実験環境

今回の実験環境は，下の表に示す．

名前	説明
デバイス	Dell XPS 12
OS	Windows 7
CPU	I7 – 3537U
Memory	8GB
File Size	1,290,240 byte (1.23MB)

実験結果

今回の実験の評価項目は３つがある．CG-SSS の秘密分散時間と秘密復元時間，ファイルの読み込む時間とファイルの書き込む時間，我々の提案の追加時間と総時間の割合である．

実際にファイルを CG-SSS を行う時，一つのファイルと単位でデータ秘密として秘密分散と秘密復元を行うことではなく，一定ビット長のブロックサイズでファイルを分割し，分割されたブロックで秘密分散と秘密復元する．今回の実験は，ブロックサイズが 256 ビットずつで実験項目を評価する．

図６．分散時間と復元時間

図６から見ると，ブロックサイズが 3584 以下の時，ブロックサイズの増加によって，分散時間と復元時間同時に減少している．その反面，ブロックサイズが 3584 ビット以上だったら，ブロックサイズの増加によって，分散時間は減少し続っているが，復元時間は逆に増加し始める．

なぜそのような現象になる理由は，分散時間と復元時間に影響された要素は主に２つのパラメータである．

分散時間には，毎回多項式計算時間 t_r と多項式計算の実行回数 r で影響される．分散時間は $t_{分}=t_r \times r$ で表示できる．毎回多項式計算時間 t_r は，ブロックサイズの増加によって，計算空間が増加するので，多項式計算が増えせる．多項式計算の実行回数 r は，ブロックサイズの増加によって，実行回数が減少される． t_r が分散時間の影響よりが r 大きいので，$t_{分}$はブロックサイズの増加によって，ずっと減少している．

復元時間には，毎回ラグランジュ補間計算時間 t_r と実行回数 r で影響される．復元時間は $t_{復}=t_r \times r$ で表示できる．ブロックサイズの増加によって，毎回ラグランジュ計算空間が増加して，毎回時間 t_r が増加される．ブロックサイズの増加によって，実行回数 r が減少される．ブロックサイズが 3584 ビット以下の時，ブロックサイズの増加による総計算は，r が復元時間の影響より t_r が大きいので，ブロックサイズが 3584 以上の時，r が分散時間の影響よりが t_r 大きい．

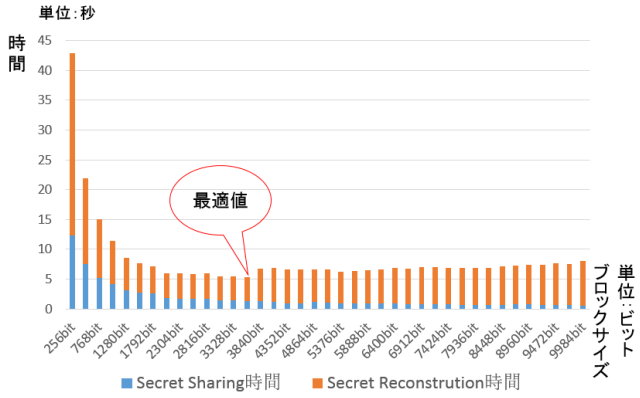

図７．分散時間と復元時間合計

図７は，分散時間と復元時間の合計図である．図７から見ると，合計時間は，ブロックサイズが 256 ビットから 3584 ビットまでに，ブロックサイズの増加によって，合計時間は減少しているが，3584 ビット以後になると，合計時間は逆に増加する傾向が見られる．そして，合計時間が一番少ないところは 3584 ビットであり，ブロックサイズの最適値は，3584 ビットになる．合計時間 $t_{合}=t_{分}+t_{復}$で表示され，分散時間より復元時間ほうが合計時間に影響が大きいことが分ける．

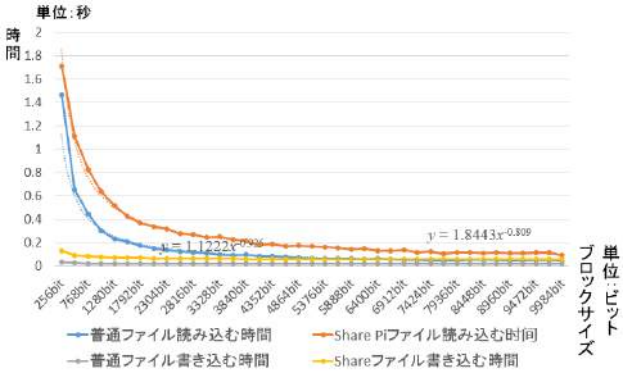

図８．ファイル操作時間

図８は，ブロックサイズによって，ファイルの読み込む時間と書き込む時間の変化図である．この図から見ると，ブロックサイズの増加によって，ファイルの読み込む時間は，多項式で減少していく一方で，ファイルの書き込む時間がほとんど変わらないことが明らかになった．その原因は，ファイルを読み込む時，データ処理のために，データタイプの変換が必要なので，使うライブラリによって，読み込む時間が変わる．今回は使っている NTL というライブラリは，業界で頻繁に使われている物

である．図８から見ると，ブロックサイズが小さいビットを選んだ時，読み込む時間は極端に大きいが，ブロックサイズが一定程度を到達すれば，ある数値に収束することが分かる．そこで，本稿では，推奨のブロックサイズは 4096 ビットより長いビット長である．

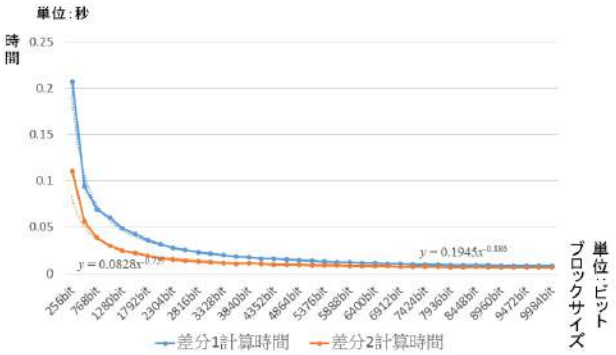

図９．差分計算時間

図９は，CG-SSS の差分計算時間である．CG-SSS の差分時間は，式 $p = s - q$ と式 $s = p + q$ の計算時間である．図９から見ると，ブロックサイズの増加により，差分計算時間が多項式で減少していく．そして，ブロックサイズが一定ビット長を超えると，計算時間はある数値に収束することが明らかになった．

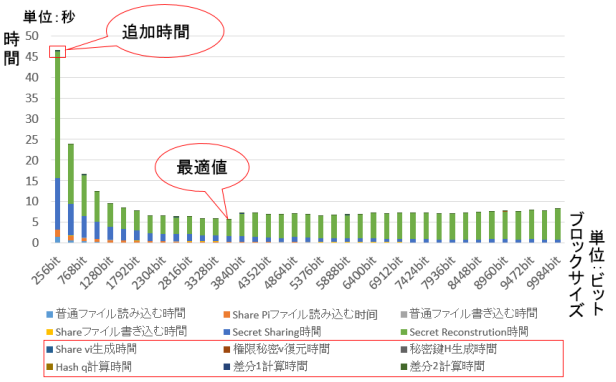

図１０．CG-SSS 総計算時間

図１０は，CG-SSS スキームの総実装時間である．図１０から見ると，ブロックサイズが 3584 ビット以下の時，ブロックサイズの増加により，総実装時間が減少する一方で，3584 ビット以後だったら，総実装時間が逆に増加していく．故に，ファイルを分けるブロックサイズの最適値は，3584 ビットになる．また，図１０の赤い四角の枠内の時間は，CG-SSS が SSS と比べ，追加時間である
．追加時間が総実装時間の割合は，図１０が示されているように，棒の先にある見えないほぼ部分だけである．すると，我々は，CG-SSS と SSS の実装時間はほぼ同じと言える．

8　まとめ

本稿では，我々はデータ秘密を復元する条件を単なるデータシェアの数に依存ではなく，プロバイダの承諾の数を復元の条件も付加する２つ閾値(t,m)-(k,n)を基づいたグループ横断秘密分散法を提案した．我々の提案のメリットは，一つのプロバイダに閾値個以上のシェアを預けることができ，一つのプロバイダのみ秘密を復元されることである．また，我々の提案は，実装が容易され，別の方式に比べ，少ないオーバーヘッドが発生される．

謝辞

第二著者に関し本研究は部分的に JSPS 科研費 15K00029 の助成を受けております．
第四著者に関し本研究は部分的に JSPS 科研費 15K00186 の助成を受けております．
第五著者に関し本研究は部分的に JSPS 科研費 15H02711 の助成を受けております．

参考文献

[1] Shamir, A. 1979. How to share a secret. In Communications of the ACM. 22.11 (Nov. 1979), 612-613.

[2] The World’s First High-speed Secret Sharing Engine for OpenStack Swift. DOI=http://www.ntt.co.jp/news2015/1505e/150518a.htm

[3] Smith, G., Boreli, R., and Kaafar, M. A. 2013. A Layered Secret Sharing Scheme and its Application to OSN Groups. Mobile and Ubiquitous Systems: Computing, Networking, and Services. 131 (Sep. 2014), 487-499.

[4] Zhang, Z., Chee, Y. M., Ling, S., Liu, M., and Wang, H. 2012. Threshold changeable secret sharing schemes revisited. Theoretical Computer Science, 418 (Feb. 2012), 106-115.

[5] Martin, K. M., Pieprzyk, J., Safavi-Naini, R., and Wang, H. 1999. Changing thresholds in the absence of secure channels. In Information Security and Privacy, 1587 (July 2001), 177-191.

[6] Beimel, A. 1996. Secure schemes for secret sharing and key distribution. Diss. Technion-Israel Institute of technology, Faculty of computer science. 27-31.

[7] Desmedt, Y., and Jajodia, S. 1997. Redistributing secret shares to new access structures and its applications. Technical Report ISSE. 148, TR-97-01, George Mason University.

[8] Herzberg, A., Jarecki, S., Krawczyk, H., and Yung, M. 1995. Proactive secret sharing or: How to cope with perpetual leakage. Advances in Cryptology—CRYPT0’95. 963,(Jul. 2001). 339-352.

[9] Pang, L., Li, H., Yao, Y., and Wang, Y. 2008. A verifiable (t, n) multiple secret sharing scheme and its analyses. In Electronic Commerce and Security. 2008 International Symposium on. IEEE, 22-26.

[10] Tompa, M., and Woll, H. 1989. How to share a secret with cheaters. Journal of Cryptology 1,3, 133-138.

[11] Ostrovsky, R., and Yung, M. 1991. How to withstand mobile virus attacks. In Proceedings of the tenth annual ACM symposium on Principles of distributed computing. PODC '91. ACM, New York, NY, 51-59.