シェア形式の変換による柔軟な秘密分散鍵管理方式

須藤 弘貴[1,a)]　菊池 亮[1]

概要：デジタルウォレットが実用化される中で、鍵管理手法の重要性が認識されるようになった。中でも、安全性を高める手法として鍵を秘密分散して保持する手法が注目されている。これまで、閾値署名と呼ばれる、鍵を秘密分散したまま生成・署名する技術が盛んに研究されてきたが、現状では冗長性と実行速度にはトレードオフがあり、実用上は最も処理コストの低い $(2,2)$ 閾値秘密分散で分散される閾値署名が実装されることがある。この場合、冗長性がないため耐障害性はなく、端末故障や紛失等で鍵シェアを 1 つ失った場合鍵を復旧することが出来なくなってしまう。そこで、本研究では 2 者間秘密分散 ((2,2) 閾値秘密分散) のシェアと冗長性のある異なる秘密分散方式との間の変換プロトコルにより柔軟に冗長性や復旧条件の変更が可能な鍵管理方式を提案する。

キーワード：閾値署名, 秘密分散, 鍵管理, デジタルウォレット, シェア変換

Flexible Secret-Sharing Based Key Management Scheme via Secure Share Conversion

Hiroki Sudo[1,a)]　Ryo Kikuchi[1]

Abstract: As digital wallets begin to see real-world use, the importance of robust key management techniques has become widely recognized. In particular, schemes that store private keys as secret shares to enhance security have attracted significant attention. To date, threshold signature schemes—which perform key generation and signing on secret shares without reconstructing the private key—have been extensively studied. However, these schemes typically entail a trade-off between redundancy and execution speed, and in practice, implementations sometimes adopt the lowest-overhead, non-redundant two-party threshold signature schemes. In such cases, the absence of redundancy results in no fault tolerance: if one key share is lost due to device failure or loss, the master key cannot be recovered. To address this limitation, we propose a key management scheme that enables flexible adjustment of redundancy levels and recovery conditions through a conversion protocol between two-party secret sharing ((2,2) threshold secret sharing) and alternative redundant secret sharing schemes.

Keywords: threshold signature, secret sharing, key management, digital wallet, share conversion

1. はじめに

1.1 背景

近年、欧州および日本において、身分証明や支払情報などのデジタル資格・資産を電子的に保管・利用するための仕組みであるデジタルウォレットの普及が急速に進展している。ビットコインのようなデジタル通貨から始まり、EU では EU Digital Identity Wallet の構想の下、国境を越えて公共サービスや民間サービスに安全かつ円滑にアクセス可能な基盤の整備が進められており、将来的には各加盟国において、この共通基盤を用いて利用者はスマートフォン上で身分証明、資格証明、支払い等を一元的に管理できる環境を目指している。日本においても、マイナンバーカードや運転免許証のデジタル化、ならびにモバイル決済サービスの高度化に伴い、デジタルウォレットが日常生活に浸透しつつある。

[1] NTT 社会情報研究所
NTT Social Informatics Laboratories
[a)] hiroki.sudo@ntt.com

このような潮流の中で、従来はサービス事業者が保有・管理することが多かった署名鍵を、利用者が自らのデバイスやウォレット内で直接管理するモデルが拡大している。署名鍵の管理は、セキュリティおよびプライバシー保護の観点から極めて重要であり、漏洩や不正利用はなりすましや資産の不正移転に繋がり、また紛失もサービスへのアクセス不能や資産の喪失といった深刻な問題を招き得る。

そのため署名鍵の安全性と可用性が重要視されており、秘密分散を用いた鍵管理手法が注目されている．秘密分散でも特に代表的な (t, n) 閾値秘密分散では，秘密鍵を n 個のシェアに分割し，そのうち t 個を集めることで元の秘密を復元できるため、$t-1$ 個までの紛失に耐えることができる。また、閾値署名と呼ばれる、署名鍵を秘密分散されたまま署名を生成する技術を用いることにより，署名鍵を一度も復元せずに署名生成が可能である．こうした方式は，単一障害点をなくしセキュリティを高める手法として有望視されており，多くの研究が進められてきた [1], [2], [3], [4]

しかしながら，実用面では冗長性と処理効率の間にトレードオフが存在し，多パーティ数を想定した方式では冗長性を高く設定できるものの，計算コストや通信コストが増加するという課題がある．そのため，実際のサービスでは最も処理コストの低い $(2, 2)$ 閾値秘密分散で分散される閾値署名が実装されることがある。この場合，秘匿性はあるが端末故障や紛失で一方のシェアが失われると署名鍵を再構成できず，デジタル情報や資産にアクセスできなくなる危険がある．

本研究では，そのような $(2, 2)$ 閾値秘密分散を基にした閾値署名が実装されているケースにおいて、$(2, 2)$ 閾値秘密分散のシェアと冗長性を持つ他の秘密分散（例：Shamir の (t, n) 方式）との間でシェア形式を相互に変換するプロトコルや、紛失時にシェアを復元するプロトコルを提案することで、既に実装されている軽量な $(2, 2)$ 閾値秘密分散を利用した閾値署名はそのまま活かしつつ，冗長な構成に切り替えて紛失時や漏洩時に署名鍵を復旧できる方法を提供する。

1.2 貢献

本研究では，シェア変換プロトコル (再分散プロトコル) により冗長性を柔軟に切り替え可能な鍵管理方式を提案する．また，紛失した $[\![a]\!]_j$ を乱数成分を変えずに復旧するプロトコルを提案する。このプロトコルでは、紛失した $[\![a]\!]_j$ そのもの、すなわちシェアの乱数成分を含めて全く同一のものが復旧される。そのため、再分散と異なり紛失していないパーティのシェアは更新する必要が無い。

1.3 関連研究

閾値署名

閾値署名は，署名鍵を (t, n) 閾値秘密分散したまま鍵生成・署名を行う技術である．署名鍵の再構成を伴わないため単一障害点を回避でき，可用性・耐漏洩性の観点から広く研究されてきた．これまでに，ECDSA の閾値化を実現した手法として Lindell らの研究 [1] や Gennaro らの手法 (GG18，GG20) [2], [3] などが知られている．また，Schnorr 署名をベースとした FROST などの閾値化も提案されている [4].

Social Recovery

提案手法と同様に紛失漏洩時の対策を行う技術として Social Recovery が研究されている．Social Recovery は，ユーザが選出したガーディアン集合の協力により鍵を回復する手法である．ANARKey [5] は，公開点（あるいはそのハッシュ）を “掲示板” に保持しつつ，ガーディアンは自分自身の秘密鍵以外のバックアップ状態を恒常的に保持しないため，スケール時の保管コスト増を回避できる．しかし，ANARKey では一度元の秘密鍵を復元する方法であり，本研究のように秘密分散のままシェアを復旧したり再分散することは想定されていない.

シェア再生成

シェアの再生成プロトコルとして Herzberg らの手法が提案されている [6]．この手法は，復旧対象のパーティ r の Shamir の多項式における座標で $f(r) = 0$ となる制約付きランダム多項式を各サーバが生成・加算するプロトコルであり，2 ラウンド・$O(nt)$ 通信量で (t, n)-Shamir 秘密分散シェアの復旧を行うことが可能である．しかし，Herzberg らの手法はランダム多項式生成が r に依存するために，事前に生成しておくことができない，ランダム多項式生成時に PRG のシード共有による非対話化 [7] が可能か示されていないといった課題があった．本研究で提案する再生成プロトコルでは独立した乱数シェア生成を用いることで，同等の通信コストを保ちつつ，事前計算や PRG のシード共有による通信ラウンドの削減が可能である．

2. 準備

2.1 攻撃者モデル

本稿では，敵対者の行動として passive および active モデルを想定する．passive な敵対者はプロトコルに従いつつ、受信したメッセージから追加の情報を取得しようとする．一方，active な敵対者はプロトコルから逸脱することを含む、任意の行動が可能である．

2.1.1 秘密分散

秘密分散とはデータを複数の値に分けて複数パーティに分散する暗号化手法である．本研究では (t, n) 閾値秘密分散によりデータを暗号化する．(t, n) 閾値秘密分散とは，データを n 個のランダムな値 (シェアと呼ばれる) に分割して，t 個以上のシェアを集めると元のデータを復元でき，t 個未満のシェアからは元データの情報を得られないような性質を持つ秘密分散法である．本研究では具体的には

Shamir 秘密分散 [8] を用いる．$\mathbb{Z}_p$ 上の秘密分散は $[\![a]\!]$ と表記する．秘密値ではない，全サーバーで共通して知っていてよい値は公開値と呼ぶこととする．

また、Shamir 秘密分散において t 人の $\mathcal{P}$ が持つシェアから、他のパーティのシェアを生成するアルゴリズムを

$$\{[\![a]\!]_j\}_{P_j \notin \mathcal{P}} \leftarrow \mathsf{ShareSimShamir}(\{[\![a]\!]_i\}_{P_i \in \mathcal{P}}) \qquad (1)$$

と書く。このアルゴリズムは具体的には t 個の座標から P_j の座標をラグランジュ補間で計算すればよく、決定的アルゴリズムである。

2.1.2 Feldman's Verifiable Secret Sharing

Verifiable Secret Sharing（VSS）は，分散者（ディーラー）が配布した各シェアが正しいことを，各参加者が（復元手順を実行せずに）検証できるようにする仕組みである．なお、ここでの “シェアが正しい” とは、正規の分散アルゴリズムによって生成されうるシェアであることを意味し、特に Shamir の (t, n) 閾値秘密分散においては、各シェアが何かしらの $t-1$ 次多項式の点であることに相当する。

Shamir の (t, n) 閾値秘密分散法 [8] を基礎とした VSS の一つとして、Feldman [9] による非対話型の検証可能秘密分散方式（Feldman's VSS）が知られている．この方式では，まず通常の Shamir 秘密分散と同様に、ディーラーは秘密値 s を定数項とする次数 $t-1$ の多項式 $f(x) = a + r_1 x + \cdots + r_{t-1} x^{t-1}$ を生成し，各参加者 P_i にシェア $[\![a]\!]_i = f(i)$ を配布する。Feldman's VSS ではさらに，各係数の乱数に対するコミットメントとして $g^a, g^{r_1}, \ldots, g^{r_{t-1}}$ を公開する。(g は巡回群 $\mathbb{G}$ 上の生成元) 各参加者は自身のシェアについて

$$g^{[\![a]\!]_i} \stackrel{?}{=} g^a \prod_{j=1}^{t-1} (g^{r_j})^{\chi_i}$$

が成り立つことを確認することで，シェアが多項式に基づき正しく生成されたことを検証できる．ただし，χ_i は Shamir の多項式における P_i の x 座標．これにより，ディーラーが不正なシェアを配布したときに各参加者は検知することができる。この処理を以下のように書く。

$$\texttt{true/false} \leftarrow \mathsf{Verify}_{\mathsf{VSS}}([\![a]\!]_i, g^a, \{g^{r_i}\}_{1 \le i \le t-1})$$

2.2 閾値署名

閾値署名方式は鍵生成プロトコルと署名生成プロトコルからなる．鍵生成プロトコルでは，n パーティが協力して (t, n) 閾値秘密分散に基づく署名鍵を生成する．この際，不正なシェアや改ざんされたシェアを検出するために VSS 等を用い，鍵シェアとともに配布される．署名生成プロトコルでは，各パーティの鍵シェアを用いて署名鍵を復元することなく署名を生成する．

本研究では代表的な閾値署名方式である GG18 [2] や GG20 [3] で改ざん検知に用いられる Feldman's VSS を想定したプロトコル設計を行う．

Protocol 1 再分散 [11]

Functionality: $[\![a]\!] \leftarrow \textsc{ReshareShamir}([\![a]\!], \mathcal{P}_{\mathsf{in}}, \mathcal{P}_{\mathsf{out}})$

Input: $[\![a]\!]_i$, and the sets of parties $\mathcal{P}_{\mathsf{in}}$ and $\mathcal{P}_{\mathsf{out}}$, where $\mathcal{P}_{\mathsf{in}}$ denote the set of parties holding the input shares, and $\mathcal{P}_{\mathsf{out}}$ denote the set of parties that receive the output shares. $|\mathcal{P}_{\mathsf{in}}| \ge t$

Output: $[\![a]\!]$

1: **Each** $P_i \in \mathcal{P}_{\mathsf{in}}$ **do**
2: $\quad d_i := \lambda_i [\![a]\!]_i$, where λ_i is a lagrange coefficient.
3: $\quad$ Share d_i via $(t', |\mathcal{P}_{\mathsf{out}}|)$-Shamir secret sharing; obtain $\langle d_i \rangle$.
4: $\quad P_i$ sends $\langle d_i \rangle$ to all parties in $\mathcal{P}_{\mathsf{out}}$.
5: **Each** $P_i \in \mathcal{P}_{\mathsf{out}}$ **do**
6: $\quad [\![a]\!]_i := \sum_{j | P_j \in \mathcal{P}_{\mathsf{in}}} \langle d_j \rangle_i$
7: Output $[\![a]\!]$

2.2.1 秘密分散上でのローカル演算

Shamir 秘密分散のような準同型性を持つ秘密分散法では，シェア同士の加算，公開値との加算，公開値との乗算，符号反転については各パーティが持つシェアのみを用いてローカルで計算できる．それぞれの演算について以下のように表記する．

- 加算: $[\![x]\!] + [\![y]\!] = [\![x + y]\!]$
- 公開値との加算: $[\![x]\!] + y = [\![x + y]\!]$
- 公開値との乗算: $[\![x]\!] \cdot y = [\![x \cdot y]\!]$
- 符号反転: $-[\![x]\!] = [\![-x]\!]$

2.3 乱数シェア生成

Shamir 秘密分散の乱数シェア生成を $\textsc{RandShamir}()$ と書くこととする．passive 攻撃者に対して安全な場合の乱数シェアを各々生成して分散しあう自明な方法では $O(tn)$，1 ラウンド通信を必要とする．active 攻撃者に対して安全な方法としては例えば Damgard らの手法 [10] が知られており，この手法の場合 $O(n^2)$ 通信量，1 ラウンド通信を必要とする．また，擬似乱数の鍵を事前に共有しておくことでローカル処理のみで実現することも可能である [7].

2.4 再分散プロトコル

再分散プロトコルは，t パーティが協力して，秘密値を変えずに新たなシェアを生成するプロトコルである．Desmedt らによる手法が知られている [11] また，シェアのリフレッシュ (乱数成分を変える) だけでなく，秘密分散方式を変える (例えば，閾値 t, パーティ数 n) ことも可能である．プロトコル 1 に [11] を Shamir 秘密分散に具体化した場合の再分散プロトコルを示す.

上述の再分散プロトコルでは，$P_i \in \mathcal{P}_{\mathsf{in}}$ が不正なシェアを生成した場合、他のパーティはそのシェアを検証できない。これを防ぐために、Feldman の VSS を用いて、各パーティが生成したシェアが正しいことを検証する手法が Wong らによって提案されている [12]．プロトコル詳細はプロトコル 2 に示す。

Protocol 2 改ざん検知あり再分散 [12]

Functionality: $[\![a]\!] \leftarrow \text{ReshareShamir}([\![a]\!], \mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}})$

Input: $[\![a]\!]_i$, parties $\mathcal{P}_{\text{in}}$, and $\mathcal{P}_{\text{out}}$,and commitments of VSS g^a and $\{g^{r_i}\}_{1\le i\le t-1}$.

Output: $[\![a]\!]$

Each $P_i \in \mathcal{P}_{\text{in}}$ **do**
  $d_i := \lambda_i [\![a]\!]_i$, where λ_i is a lagrange coefficient.
  P_i generates $\langle d_i \rangle$ by performing $(t', |\mathcal{P}_{\text{out}}|)$-Shamir secret sharing.
  Share d_i with Feldman's VSS and compute $\langle d_i \rangle$, g^{d_i}, and $\{g^{\mu_{i,j}}\}_{1\le j\le t-1}$.
  P_i sends $\langle d_i \rangle$, $\{g^{\mu_{i,j}}\}_{1\le j\le t-1}$ to $\mathcal{P}_{\text{out}}$.
Each $P_j \in \mathcal{P}_{\text{out}}$ **do**
  (Broadcasts $\perp$ and aborts if $g^a, \{g^{r_i}\}_{1\le i\le t-1}$ received from all parties are not all identical)
  Each $P_i \in \mathcal{P}_{\text{in}}$ **do**
    if $\text{Verify}_{\text{VSS}}(\langle d_i \rangle_j, g^{d_i}, \{g^{\mu_{i,j}}\}_{1\le j\le t-1}) =$ `false` **then**
      P_j Broadcasts $\perp$ and aborts
  Broadcasts $\perp$ and aborts if $g^a \neq \prod_{j=0}^{k-1} (g^{d_j})^{\chi_j}$
  $[\![a]\!]_i := \sum_{j\in\mathcal{P}_{\text{in}}} [\![d_j]\!]_i$
Output $[\![a]\!]$

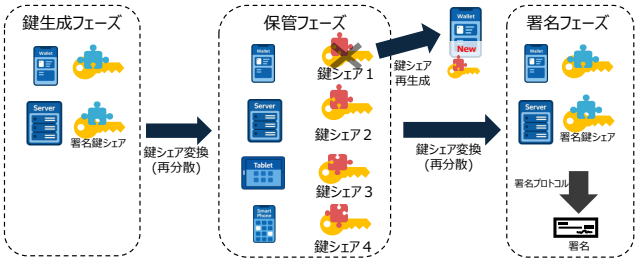

図 1 システム概観.

Fig. 1 System Overview

3. 提案手法

3.1 提案システム概観

本研究では，既に実装されている軽量な $(2,2)$ 閾値秘密分散を利用した閾値署名はそのまま活かしつつ，冗長な構成に切り替えて紛失時や漏洩時に署名鍵を復旧できるシステムを提案する。図 1 に提案手法の全体システム概観を示す。

まず，鍵生成フェーズでは閾値署名の鍵生成プロトコルにより，鍵シェアを生成する．その後，バックアップ端末群と協調して生成された鍵シェアから再分散プロトコルによる鍵シェア変換を行い，冗長性のある秘密分散形式に変換，バックアップ端末群にバックアップ用鍵シェアを格納する．鍵シェアが紛失等により失われた場合には，後述する再生成プロトコルにより，失われた鍵シェアを復旧する．また，再分散プロトコルにより鍵のリフレッシュも行うことができる．署名時には，バックアップ端末群に格納された鍵シェアから署名用鍵シェアに変換して署名を行う．

Protocol 3 基本的な再生成

Functionality: $[\![a]\!] \leftarrow \text{Regenerate}(\{[\![a]\!]_i\}_{P_i\in\mathcal{P}_{\text{in}}}, \mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}})$

Input: Shares $\{[\![a]\!]_i\}_{P_i\in\mathcal{P}_{\text{in}}}$ of t parties specified by $\mathcal{P}_{\text{in}}$, $\mathcal{P}_{\text{in}}$, and $\mathcal{P}_{\text{out}}$, where $\mathcal{P}_{\text{in}}$ denote the set of parties holding the input shares, and $\mathcal{P}_{\text{out}}$ denote the set of parties to be recovered.

Output: $[\![a]\!]$, where each $[\![a]\!]_i$ for $P_i \in \mathcal{P}_{\text{in}}$ is the same as the input share

Select P_ℓ from $\mathcal{P}_{\text{in}}$ ▷ E.g., the smallest index in $\mathcal{P}_{\text{in}}$
$[\![r]\!] \leftarrow \text{RandShamir}()$
Each $P_i \in \mathcal{P}_{\text{in}}$ **do**
  $[\![b]\!]_i = [\![r]\!]_i - [\![a]\!]_i$
Each $P_i \in \mathcal{P}_{\text{in}} \setminus \{P_\ell\}$ **do**
  Sends $[\![b]\!]_i$ to P_ℓ
Each P_ℓ **do**
  $\{[\![b]\!]_j\}_{P_j\in\mathcal{P}_{\text{out}}} \leftarrow \text{ShareSimShamir}(\{[\![b]\!]_i\}_{P_i\in\mathcal{P}_{\text{in}}})$
  Sends $[\![b]\!]_j$ to P_j for $P_j \in \mathcal{P}_{\text{out}}$
Each $P_j \in \mathcal{P}_{\text{out}}$ **do**
  $[\![a]\!]_j = [\![r]\!]_j - [\![b]\!]_j$
Output $[\![a]\!]$

3.2 再生成プロトコル

本節では、あるパーティ P_j がそのシェア $[\![a]\!]_j$ を紛失してしまった際に、t パーティが協力し、紛失した $[\![a]\!]_j$ を復旧するプロトコルを提案する。このプロトコルでは、紛失した $[\![a]\!]_j$ そのもの、すなわちシェアの乱数成分を含めて全く同一のものが復旧される。そのため、再分散と異なり紛失していないパーティのシェアは更新する必要が無い。

まず基本的な再生成プロトコルを提案し、その後に検証可能な再生成プロトコルを提案する。既に述べた通り、本稿では紛失を攻撃者の挙動の一環（シェアの不正な削除）と捉えているため、$t-1$ 個のシェアが紛失していて復旧する場合、攻撃者が高々 $t-1$ 人という前提と合わせてシェアの再生成に協力するパーティは全て honest であると考えられる。Honest であれば一度復元する方法も可能であるが、復元せず誰も署名鍵を見ない方法もそれほどコストがかからず可能なため、（復元せずに）再生成する方法としてプロトコル 3 を提案する。また，リーダーを経由せず再生成対象のパーティに直接メッセージを送信するバリエーションを 4 に示す。この場合，リーダーでメッセージを集約しないため再生成対象パーティの数が多い場合に総通信量が増大するが，ラウンド数を 1 回減らすことが出来る．

一方で、紛失したシェアの数が $t-1$ より少ない場合では、シェアの再生成に協力するパーティに攻撃者が含まれている場合がある。このような場合、再生成に協力するパーティによる改ざんを検知する必要がある。再生成に協力するパーティが t パーティであり、且つその中に攻撃者が含まれ得る場合、honest なパーティのシェアだけでは秘密が確定しないため、単純な Shamir のシェアのみでは改ざんを検知することが難しい。そのため何かしらのチェック用の値が必要となる。本稿ではこれを便宜的に Commitment と呼ぶ。なお、もし再生成プロトコルを流用して分散する

Protocol 4 基本的な再生成 (リーダーを置かない場合)

Functionality: $[\![a]\!] \leftarrow \text{REGENERATE}(\{[\![a]\!]_i\}_{P_i \in \mathcal{P}_{\mathsf{in}}}, \mathcal{P}_{\mathsf{in}}, \mathcal{P}_{\mathsf{out}})$

Input: Shares $\{[\![a]\!]_i\}_{P_i \in \mathcal{P}_{\mathsf{in}}}$ of t parties specified by $\mathcal{P}_{\mathsf{in}}$, and the set of parties $\mathcal{P}_{\mathsf{out}}$

Output: $[\![a]\!]$, where each $[\![a]\!]_i$ for $P_i \in \mathcal{P}_{\mathsf{in}}$ is the same as the input share

$[\![r]\!] \leftarrow \text{RANDSHAMIR}()$
Each $P_i \in \mathcal{P}_{\mathsf{in}}$ **do**
  $[\![b]\!]_i = [\![r]\!]_i - [\![a]\!]_i$
Each $P_i \in \mathcal{P}_{\mathsf{in}}$ **do**
  Sends $[\![b]\!]_i$ to $P_j \in \mathcal{P}_{\mathsf{out}}$
Each $P_j \in \mathcal{P}_{\mathsf{out}}$ **do**
  $[\![b]\!]_j \leftarrow \mathsf{ShareSimShamir}(\{[\![b]\!]_i\}_{P_i \in \mathcal{P}_{\mathsf{in}}})$
  $[\![a]\!]_j = [\![r]\!]_j - [\![b]\!]_j$
Output $[\![a]\!]$

パーティの追加に使う場合も同様のことが起きる。すなわち、追加するパーティのシェアを再生成プロトコルで生成するような場合、追加されるパーティはシェアを持っていないのが正常であるので、協力するパーティ内に攻撃者が存在する可能性があり、同様に改ざんを検知できるような仕組みが必要である。

本稿では、GG18[2] や GG20[3] 等、閾値署名においてよく用いられている Feldman's VSS の Commitment である $g^a, \{g_i^r\}_{1 \le i \le t-1}$ が、署名鍵のシェアに加えて公開されているという前提の下で、攻撃者の改ざんが検知できる再生成プロトコルであるプロトコル 5 を提案する。また，改ざん検知ありについても同様にリーダーを置かない方法を取ることができ，この場合の詳細はプロトコル 6 に示す．なお、Feldman's VSS のコミットメント $g^a, \{g_i^r\}_{1 \le i \le t-1}$ が公開掲示板で公開されるなど、改ざんされていない形で誰でもアクセス可能な場合、括弧書きの処理は省略することができる。一方でプロトコル 5 では各パーティがコミットメントを各自保持しているという前提で記載している。この場合、再復旧してもらうパーティはシェアと同様にコミットメントも紛失し、かつ攻撃者は自分の持つコミットメントに対して改ざんすることもあり得るという状況のため、相互にコミットメントを送りあい同一性チェックを行うことで正しいコミットメントを再配布している。

また、復旧に協力するパーティのうち、honest なパーティが t 人含まれていれば、プロトコル 5 での commitment のような計算量的なツールの存在を仮定せず、再復旧で改ざんを検知できる。そのようなプロトコルをプロトコル 7 で提案する。本プロトコルでは、復旧に協力するパーティの中で honest なパーティ数が少なくとも t 人必要である。ここで用いる CheckConsistency は、t 個のシェアから他のシェアをラグランジュ補間で生成し受け取ったシェアと一致すれば 1 を出す処理で、受け取ったシェアが $t-1$ 次多項式上に乗っているならば 1, そうでなければ 0 が出力される。少なくとも t パーティが正しくシェアを渡している

Protocol 5 改ざん検知あり再生成

Functionality: $[\![a]\!] \leftarrow \text{REGENERATE}([\![a]\!], \mathcal{P}_{\mathsf{in}}, \mathcal{P}_{\mathsf{out}}, g^a, \{g^{r_i}\}_{1 \le i \le t-1})$

Input: Shares $\{[\![a]\!]_i\}_{P_i \in \mathcal{P}_{\mathsf{in}}}$ of t parties specified by $\mathcal{P}_{\mathsf{in}}$, the set of parties $\mathcal{P}_{\mathsf{out}}$, and commitments of VSS g^a and $\{g^{r_i}\}_{1 \le i \le t-1}$

Output: $[\![a]\!]$, where each $[\![a]\!]_i$ for $P_i \in \mathcal{P}_{\mathsf{in}}$ is the same as the input share

Select P_ℓ from $\mathcal{P}_{\mathsf{in}}$
$[\![r]\!] \leftarrow \text{RANDSHAMIR}()$
Each $P_i \in \mathcal{P}_{\mathsf{in}}$ **do**
  $[\![b]\!]_i = [\![r]\!]_i - [\![a]\!]_i$
Each $P_i \in \mathcal{P}_{\mathsf{in}} \setminus \{P_\ell\}$ **do**
  Sends $[\![b]\!]_i$ to P_ℓ
  (Sends $g^a, \{g^{r_i}\}_{1 \le i \le t-1}$ to $P_j \in \mathcal{P}_{\mathsf{out}}$)
Each P_ℓ **do**
  $\{[\![b]\!]_j\}_{P_j \in \mathcal{P}_{\mathsf{out}}} \leftarrow \mathsf{ShareSimShamir}(\{[\![b]\!]_i\}_{P_i \in \mathcal{P}_{\mathsf{in}}})$
  Sends $[\![b]\!]_j$ to P_j for $P_j \in \mathcal{P}_{\mathsf{out}}$
  (Sends $g^a, \{g^{r_i}\}_{1 \le i \le t-1}$ to $P_j \in \mathcal{P}_{\mathsf{out}}$)
Each $P_j \in \mathcal{P}_{\mathsf{out}}$ **do**
  (Broadcasts $\perp$ and aborts if $g^a, \{g^{r_i}\}_{1 \le i \le t-1}$ received from all parties are not all identical)
  $[\![a]\!]_j = [\![r]\!]_j - [\![b]\!]_j$
  if $\mathsf{Verify}_{\mathsf{VSS}}([\![a]\!]_i, g^a, \{g^{r_i}\}_{1 \le i \le t-1}) = \texttt{false}$ **then**
    Broadcasts $\perp$ and aborts
Output $[\![a]\!]$ if $\perp$ is not received

Protocol 6 改ざん検知あり再生成 (リーダーを置かない場合)

Functionality: $[\![a]\!] \leftarrow \text{REGENERATE}([\![a]\!], \mathcal{P}_{\mathsf{in}}, \mathcal{P}_{\mathsf{out}}, g^a, \{g^{r_i}\}_{1 \le i \le t-1})$

Input: Shares $\{[\![a]\!]_i\}_{P_i \in \mathcal{P}_{\mathsf{in}}}$ of t parties specified by $\mathcal{P}_{\mathsf{in}}$, the set of parties $\mathcal{P}_{\mathsf{out}}$, and commitments of VSS g^a and $\{g^{r_i}\}_{1 \le i \le t-1}$

Output: $[\![a]\!]$, where each $[\![a]\!]_i$ for $P_i \in \mathcal{P}_{\mathsf{in}}$ is the same as the input share

$[\![r]\!] \leftarrow \text{RANDSHAMIR}()$
Each $P_i \in \mathcal{P}_{\mathsf{in}}$ **do**
  $[\![b]\!]_i = [\![r]\!]_i - [\![a]\!]_i$
Each $P_i \in \mathcal{P}_{\mathsf{in}}$ **do**
  Sends $[\![b]\!]_i$ to P_ℓ
  (Sends $g^a, \{g^{r_i}\}_{1 \le i \le t-1}$ to $P_j \in \mathcal{P}_{\mathsf{out}}$)
Each $P_j \in \mathcal{P}_{\mathsf{out}}$ **do**
  $[\![b]\!]_j \leftarrow \mathsf{ShareSimShamir}(\{[\![b]\!]_i\}_{P_i \in \mathcal{P}_{\mathsf{in}}})$
  (Broadcasts $\perp$ and aborts if $g^a, \{g^{r_i}\}_{1 \le i \le t-1}$ received from all parties are not all identical)
  $[\![a]\!]_j = [\![r]\!]_j - [\![b]\!]_j$
  if $\mathsf{Verify}_{\mathsf{VSS}}([\![a]\!]_i, g^a, \{g^{r_i}\}_{1 \le i \le t-1}) = \texttt{false}$ **then**
    Broadcasts $\perp$ and aborts
Output $[\![a]\!]$ if $\perp$ is not received

場合、攻撃者のシェアは一意に決定し、改ざんした場合は $t-1$ 次多項式上に乗らず検知される。

3.3 安全性

全ての提案プロトコルが実現する再生成の理想機能は、再生成に協力するパーティからシェアが入力され、それらのシェアが consistent であれば、再生成されるパーティのシェアを ShareSimShamir で生成し渡すような理想機能である。ただし Protocol 5 および 6 が実現する理想機能は、

Protocol 7 改ざん検知あり再生成（Commitment なし）

Functionality: $[\![a]\!] \leftarrow \textsc{Regenerate}([\![a]\!], \mathcal{P}_{\mathsf{in}}, \mathcal{P}_{\mathsf{out}})$

Input: Shares $\{[\![a]\!]_i\}_{P_i \in \mathcal{P}_{\mathsf{in}}}$ of m parties specified by $\mathcal{P}_{\mathsf{in}}$, and the set of parties $\mathcal{P}_{\mathsf{out}}$.

Output: $[\![a]\!]$, where each $[\![a]\!]_i$ for $P_i \in \mathcal{P}_{\mathsf{in}}$ is the same as the input share

1: $[\![r]\!] \leftarrow \textsc{RandShamir}()$
2: **Each** $P_i \in \mathcal{P}_{\mathsf{in}}$ **do**
3: $\quad [\![b]\!]_i = [\![r]\!]_i - [\![a]\!]_i$
4: **Each** $P_i \in \mathcal{P}_{\mathsf{in}}$ **do**
5: $\quad$ Sends $[\![b]\!]_i$ to $P_j \in \mathcal{P}_{\mathsf{out}}$
6: **Each** $P_j \in \mathcal{P}_{\mathsf{out}}$ **do**
7: $\quad$ If $\mathsf{CheckConsistency}(\{[\![b]\!]_i\}_{P_i \in \mathcal{P}_{\mathsf{in}}}) = 0$ **then** broadcasts $\perp$
8: $\quad \{[\![b]\!]_j\}_{P_j \in \mathcal{P}_{\mathsf{out}}} \leftarrow \mathsf{ShareSimShamir}(\{[\![b]\!]_i\}_{P_i \in \mathcal{P}_{\mathsf{in}}})$
9: $\quad [\![a]\!]_j = [\![r]\!]_j - [\![b]\!]_j$
10: Output $[\![a]\!]$ if $\perp$ is not received

追加で VSS のコミットメントを入力として consistent かどうかを理想機能で確認するものとする。提案するプロトコルは乱数シェアを利用しているため、そのような乱数シェアを生成する理想機能が存在する hybrid model で前述の再生成の理想機能を実現している。

Protocol 3 および 4 は、semi-honest adversary の $t-1$ corruption に対して安全である。各パーティからシェアを得て復元が可能であるが、復元される値は乱数により一様乱数となっており、秘密情報である a の情報は何も得ることができない。また、それ以外のパーティは自身のシェアおよび乱数シェアしか得られないため秘密の情報は得ることはできないため、前述の主張が成り立つ。

Protocol 5 および 6 は、malicious adversary の $t-1$ corruption に対して（離散対数問題が難しいという仮定のもと、with abort の意味で）安全である。まず秘匿性について考える。Protocol 3 ではリーダーの P_ℓ、Protocol 4 のどこかの $P_j \in \mathcal{P}_{\mathsf{out}}$ が corrupt されている場合を考える。このとき、高々 $t-2$ 個まで改ざんされたシェアを攻撃者は手に入れられるが、RandShamir で正しく乱数シェアが生成され少なくとも 1 パーティがその乱数シェアでマスクしているので、一様ランダムなシェアを貰うのと同等の情報しか得られない。実際、改ざんしたシェアを $[\![\tilde{b_i}]\!]$、改ざんしていない本来のシェアとの値の差分を $\delta_i = [\![\tilde{b_i}]\!] - [\![b]\!]_i$ とし、ラグランジュ補完の係数を λ_i と書くと（本来はどのシェアを計算するかによって異なるのだが簡単のために省略する）、任意の改ざんは秘密を $r - a + \sum_i \lambda_i \delta_i$ に変化させる攻撃とみなすことができる。ここで i は $\mathcal{P}_{\mathsf{in}}$ のなかで corrupt されているパーティの index であり、t 個のシェアを考えているため inconsistent なシェアにはなり得ないことに注意。$\sum_i \lambda_i \delta_i$ は攻撃者にとって既知のため新しい情報を得られていない。次に健全性（復元されたシェアは正しいものであるという性質）について考える。これは Feldman's VSS のコミットメントが正しく共有されていれば、検証式を通過しているため Feldman's VSS の性質から直ちに示される。さらに Feldman's VSS のコミットメントは少なくとも honest なパーティを 1 人含む t パーティが $\mathcal{P}_{\mathsf{out}}$ に送信し、完全一致でない場合は abort としている。以上のことから前述の主張が成り立つ。

最後に Protocol 7 は、$2t-1 \leq m$ のとき、malicious adversary の $t-1$ corruption に対して安全である。秘匿性は Protocol 6 と同様の議論が成り立つ（ただしコミットメントがないため離散対数問題が難しいという仮定は必要としない）。健全性については、もし $t-1$ シェアが改ざんされたとしても少なくとも t 個の正しいシェアが存在するため、CheckConsistency を行うことで改ざんを検知できる。また、RandShamir のあと通信は 1 ステップのため、改ざんを秘密に依存させることもできず、前述の主張が成り立つ。

3.4 計算量

各プロトコル共通して RandShamir については実装方法に依存するため，ここでは乱数シェア生成部分を除いた計算量を考察する．プロトコル 3 の場合，$\mathcal{P}_{\mathsf{in}} \setminus P_\ell$ から P_ℓ に各 1 要素を送信したのち，P_ℓ から $\mathcal{P}_{\mathsf{out}}$ 内のパーティそれぞれに 1 要素を送信するため，$|\mathbb{F}|(|\mathcal{P}_{\mathsf{in}}| + |\mathcal{P}_{\mathsf{out}}| - 1)$bits 通信，2 ラウンドの通信コストが発生する．

改ざん検知あり再生成 (プロトコル 5) の場合も公開掲示板がある場合には同様の通信コストとなる．公開掲示板がない場合，シェアの送信と並列に $\mathcal{P}_{\mathsf{in}}$ から $\mathcal{P}_{\mathsf{out}}$ への Commitment の送信を行うため，Commitment のサイズを $|\mathbb{G}|$ とすると，通信量・通信ラウンドは，$|\mathbb{F}|(|\mathcal{P}_{\mathsf{in}}| + |\mathcal{P}_{\mathsf{out}}| - 1) + |\mathbb{G}|(|\mathcal{P}_{\mathsf{in}}| \cdot |\mathcal{P}_{\mathsf{out}}|)$bits 通信，3 ラウンドとなる．

リーダーを経由しないプロトコル (基本的な再生成 (プロトコル 4)，改ざん検知あり再生成プロトコル 6，Commitment なしの改ざん検知あり再生成プロトコル 7) では，$\mathcal{P}_{\mathsf{in}}$ から $\mathcal{P}_{\mathsf{out}}$ へ直接送信するため，通信量・通信ラウンドはそれぞれ $|\mathbb{F}|(|\mathcal{P}_{\mathsf{in}}| \cdot |\mathcal{P}_{\mathsf{out}}|)$bits・2 ラウンド，$|\mathbb{F}|(|\mathcal{P}_{\mathsf{in}}| \cdot |\mathcal{P}_{\mathsf{out}}|) + |\mathbb{G}|(|\mathcal{P}_{\mathsf{in}}| \cdot |\mathcal{P}_{\mathsf{out}}|)$bits・2 ラウンド，$|\mathbb{F}|(|\mathcal{P}_{\mathsf{in}}| \cdot |\mathcal{P}_{\mathsf{out}}|)$bits・2 ラウンドとなる．

4. おわりに

本研究では，シェア変換により秘密分散切り替えを冗長性を柔軟に切り替えることのできる鍵管理方式を提案した．また，紛失したシェアの乱数成分を変えずにシェアを再生成するプロトコルを提案し，さらに改ざん検知等の拡張を行った．

参考文献

[1] Lindell, Y.: Fast Secure Two-Party ECDSA Signing, *Advances in Cryptology – CRYPTO 2017*, Springer Inter-

national Publishing, pp. 613–644 (2017).
[2] Gennaro, R. and Goldfeder, S.: Fast multiparty threshold ECDSA with fast trustless setup, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, ACM (2018).
[3] Gennaro, R. and Goldfeder, S.: One round threshold ECDSA with identifiable abort, *Cryptology ePrint Archive* (2020).
[4] Komlo, C. and Goldberg, I.: FROST: Flexible round-optimized Schnorr threshold signatures, *International Conference on Selected Areas in Cryptography*, Springer, pp. 34–65 (2020).
[5] Kate, A., Mukherjee, P., Saleem, H., Sarkar, P. and Roberts, B.: ANARKey: A New Approach to (Socially) Recover Keys, *Cryptology ePrint Archive* (2025).
[6] Herzberg, A., Jarecki, S., Krawczyk, H. and Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage, *annual international cryptology conference*, Springer, pp. 339–352 (1995).
[7] Cramer, R., Damgård, I. and Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation, *Theory of Cryptography Conference*, Springer, pp. 342–362 (2005).
[8] Shamir, A.: How to share a secret, *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613 (1979).
[9] Feldman, P.: A practical scheme for non-interactive verifiable secret sharing, *28th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 427–437 (1987).
[10] Damgård, I. and Nielsen, J. B.: Scalable and unconditionally secure multiparty computation, *Annual International Cryptology Conference*, Springer, pp. 572–590 (2007).
[11] Desmedt, Y. and Jajodia, S.: Redistributing secret shares to new access structures and its applications, Technical report, Technical Report ISSE TR-97-01, George Mason University (1997).
[12] Wong, T. M., Wang, C. and Wing, J. M.: Verifiable secret redistribution for archive systems, *First International IEEE Security in Storage Workshop, 2002. Proceedings.*, IEEE, pp. 94–105 (2002).