ブロックチェーン秘密鍵のバックアップエコシステムの提案

Proposal of Blockchain Private Key Backup Ecosystem

白濱敬也 †, 森山真光 †
Takaya Shirahama†, and Masamitsu Moriyama†
† 近畿大学大学院 総合理工学研究科
†Graduate School of Science and Engineering Research, Kindai Univ.

要旨

本稿の目的は，ブロックチェーン上で使用されている秘密鍵を，機密性と可用性に優れた状態でバックアップすることである．従来のバックアップ手法は，機密性のみに焦点を当てた設計が中心的であった．しかしながら，可用性については等閑に付されてきたことによって，バックアップからの復元が即座に行えないという問題がある．そこで本稿では，ブロックチェーン秘密鍵のバックアップエコシステムの提案を行う．まず，機密性に焦点を当て，鍵導出関数と秘密分散法を利用したバックアッププロトコルを示す．次に，可用性に焦点を当て，デジタル署名によるデータ検証プロトコルを示す．そして，主に機密性・可用性の観点より提案手法を考察する．

1. はじめに

ブロックチェーンは，資産の集中管理からの脱却という開発思想を持つ非中央集権な仕組みである．ブロックチェーン上の資産は暗号資産と呼ばれ，その所有権はデジタル署名によって証明される．そして，この署名で用いる秘密鍵の個人管理は，ブロックチェーンを利用するうえでの基本原則である．秘密鍵の喪失は暗号資産の喪失と同義であるため，バックアップの重要性が高い．BIP[1] で複数の提案がされていることからも，その重要性は明らかである．

秘密鍵のバックアップ手法は，オフライン手法とオンライン手法に二分される．管見によれば，ブロックチェーン利用者の多くはオフライン手法を実施している．スマートコントラクト[2] を搭載した初のプラットフォーム Ethereum のウォレット MetaMask[3] では，BIP39 を用いたオフライン手法が推奨されている．オフライン手法は，オンライン攻撃に遭わないため機密性が高く，常に復元が行えるわけではないため可用性が低い．一方，オンライン手法は，設計によって変わるため機密性については一概に評価できないが，ネットワークにさえ接続できれば復元が行えるため可用性が高い．そこで，機密性が高いオンライン手法が提案できれば，機密性と可用性の両方に考慮して秘密鍵をバックアップすることができると考えた．オンライン手法について一部言及している先行研究 [1] では，秘密分散法で暗号化秘密鍵を分割し，分割データをノードに共有することによる秘密鍵復元プロトコルを提唱していた．しかし，この先行研究では，データを保管していることの証明（Proof of Custody 以下 PoC）に関する言及がないことから可用性に疑問が残る．

2. 関連研究

2.1. 秘密分散法

秘密分散法（Secret Sharing Scheme）では，秘密情報 S を n 個の分散情報 (share)$W_j, j = 1, 2, \cdots, n$，に分散符号化する．(k, n) しきい値法では，任意の k 個の分散情報から秘密情報 S が復元でき，任意の $k-1$ 個の分散情報からは S が全く分からないという特徴を持つ [2].

2.2. 鍵導出関数

鍵導出関数（Key Derivation Function）は，パスワードやパスフレーズなどという秘密の値を元に鍵を導出する関数である．salt を使うことでルックアップテーブルアタックとレインボーテーブルアタックへ対抗でき，キーストレッチを使いハッシュ関数を遅くすることで，ディクショナリアタックとブルートフォースアタックへ対抗できる．この鍵導出関数の標準を策定する PHC（Password Hashing Competition）[4] では，Argon2[3] を使用することが推奨されている．

---

[1] BIP（Bitcoin Improvement Proposals）（2020/10/01 参照）`https://github.com/bitcoin/bips`
[2] ここでは，ブロックチェーン上に配備され動作するプログラム，またはその仕組みそのものを指す.
[3] METAMASK （2020/10/01 参照）`https://metamask.io`
[4] Password Hashing Competition（2020/10/01 参照）`https://password-hashing.net`

3. ブロックチェーン秘密鍵のバックアップエコシステム

以降，表 1 の表記を使用する．

表 1: 表記の説明

表記	説明
$Rand$	乱数生成関数
KDF	キー派生関数
SSS	秘密分散法
Enc	暗号化関数
$Sign$	署名関数
W	SSS で導出される分散情報の集合
w_i	W の要素 $(1 \leq i \leq n)$
m_i	w_i に対応するメタデータ
r_l	長さ l の乱数データ
pw	パスワード
$cost$	KDF でのコストパラメータ
n	SSS での分割数 $(2 \leq n \leq 255)$
k	SSS でのしきい値 $(1 \leq k \leq n)$
dk	派生キー
s	ブロックチェーン秘密鍵のシード値

表記	説明
es	暗号化されたシード値
D	バックアップデータの集合
d_i	w_i と m_i で作成するバックアップデータ
pk_i	d_i をシード値とする公開鍵
sk_i	d_i をシード値とする秘密鍵
sig_i	sk_i による署名
C	クライアントの集合
c	ある任意のクライアント
P	プロバイダの集合
$P\alpha$	非共謀プロバイダの集合 $P\alpha \subseteq P$
$P\beta$	共謀プロバイダの集合 $P\beta \subseteq P$
P_c	c が利用するプロバイダの集合
p	ある任意のプロバイダ
p_i	d_i を保管するプロバイダ $p_i \in P_c$
$\|\Gamma\|$	任意の有限集合 Γ の要素数

3.1. リレーション

提案するエコシステムには，オンラインデータの保管サービスを提供するプロバイダ P と P のサービスを利用するクライアント C，クライアント–プロバイダ間のエスクロー[5] を実現するスマートコントラクトが存在する．図 1 のように，あるクライアント c は単一のスマートコントラクトと任意のプロバイダ p を複数選択し利用する．

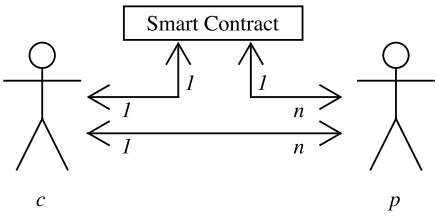

図 1: リレーションモデル

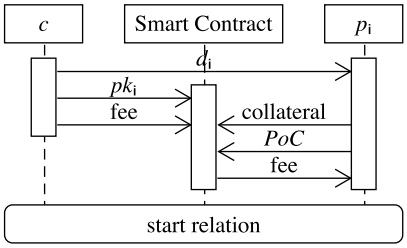

図 2: リレーション開始

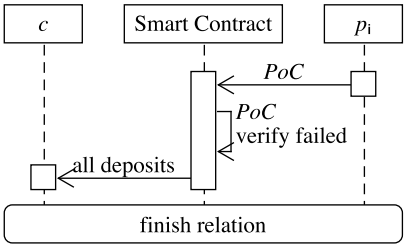

図 3: PoC 失敗

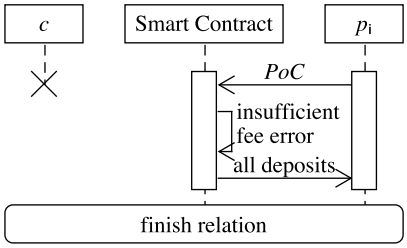

図 4: 手数料デポジット不足

クライアント–プロバイダのリレーション開始は，図 2 のようなプロセスで行われる．まず，c はバックアップデータの集合 D を作成し，D の要素 d_i をプロバイダ p_i へ送信する．次に，p_i がサービス保証金（図 3–collateral）のデポジットを，c が手数料（図 3–fee）のデポジットを行う．最後に，初めての PoC と手数料受け渡しを成功させることで，リレーションが開始する．

リレーションの継続中，p_i は一定期間毎に PoC と手数料の引き出しを行いながら，c に対してサービスを提供する．一方 c は，p_i の PoC 時に手数料が不足しないようデポジット額を維持し続ける．

[5] 商取引の際に信頼の置ける第三者．買い手と売り手を仲介し取引の安全を担保する．

リレーションの終了は，PoC もしくは手数料受け渡しにて問題が発生した場合に起こる．具体的には，p_i が PoC に失敗した場合（図 3）と PoC 成功時に手数料が不足していた場合（図 4）である．前者はすべてのデポジットが c へ渡り，後者は p_i へ渡る．

3.2. バックアップデータの作成

入力は，パスワード pw・シード値 s・秘密分散法 SSS のしきい値 k と分割数 n・鍵導出関数 KDF のコストパラメータ $cost$ である．乱数 r_{24} は自動生成するため入力は不要である．まず，式 (1) のように，KDF で pw から派生鍵 dk を得る．

$$dk \quad = \quad KDF(pw,\, r_{24},\, cost) \tag{1}$$

次に，式 (2) のように，dk と r_{24} を用いて s を Enc で暗号化し，暗号化された秘密鍵 es を得る．

$$es \quad = \quad Enc(s,\, dk,\, r_{24}) \tag{2}$$

さらに，式 (3) のように，SSS で r_{24} の分散情報の集合 W を導出する．

$$W \quad = \quad SSS(r_{24},\, n,\, k) \tag{3}$$

Algorithm 1 Backup Data Creation

Input: pw, s, k, n, $cost$
Output: es, D

$r_{24} \leftarrow Rand(24)$
$dk \leftarrow KDF(pw,\, r_{24},\, cost)$
$es \leftarrow Enc(s,\, r_{24},\, dk)$
$W \leftarrow SSS(r_{24},\, k,\, n)$
for $i = 1, \cdots, n$ **do**
  $m_i \leftarrow i \,||\, k \,||\, n \,||\, cost$
  $d_i \leftarrow w_i \,||\, m_i$
end for
$D \leftarrow \{\, d_1,\, d_2,\, \cdots,\, d_n \,\}$
return es, D

最後に，式 (4) のように，KDF で用いたコストパラメータ $cost$，SSS で用いたパラメータ n および k から，p_i に割り当てるメタデータ m_i を作成する．これを式 (5) のように w_i と結合することで，分散バックアップデータ d_i を得る．

$$\underbrace{m_i}_{8byte} \quad = \quad \{\underbrace{i}_{1byte} || \underbrace{cost}_{5byte} || \underbrace{n}_{1byte} || \underbrace{k}_{1byte}\} \tag{4}$$

$$\underbrace{d_i}_{32byte} \quad = \quad \{\underbrace{m_i}_{8byte} \;||\; \underbrace{w_i}_{24byte}\} \tag{5}$$

なお，d_i は p_i による PoC において秘密鍵 sk_i のシード値としても使用する．そのため，d_j のデータサイズは 32byte であることが要求される．データの作成過程で利用される乱数 r_{24} は 24byte であり，(k, n) しきい値法で作成する分散情報の各データサイズは元のデータサイズと等しくなることから，w_i も 24byte となる．つまり，m_i が 8byte となるような調整が必要である．

3.3. PoC

c は前もって d_i をシード値とした公開鍵 pk_i と，乱数 r_{32} をスマートコントラクトへ送信しておく[6]．PoC は以下の手順によって行われる．式 (6) のように p_i は d_i から sk_i を導出して r への署名を行う．スマートコントラクトは pk_i を用いて署名データ sig_i の検証を行う．

$$sig_i \quad = \quad Sign(sk_i,\, r_{32}) \tag{6}$$

4. 考察

4.1. 機密性

es から s を導出するためには r_{24} の復元と pw の解析が必須である．pw の解析において，$cost$ が重ければ費用対効果が低くなるため，攻撃対象にされにくい．以下は，r_{24} への攻撃について考察する．

まず，P_c に含まれない外部からの攻撃を考察する．攻撃者が r_{24} を得るためには，P_c の中から k 以上のプロバイダを攻撃しなければならない．これは現実的ではなく，十分に機密性が高いといえる．

次に，共謀攻撃を考察する．これは P_c 内に k 以上の共謀者が含まれる場合に発生する．式 (7) は共謀可能確率（P から p をランダムに n 個選ぶ際に $P\beta$ から k 以上選ぶ確率）を表す．

$$P(collusionable) = \sum_{i=k}^{n} \frac{{}_{|P\alpha|}C_{n-i} \times {}_{|P\beta|}C_i}{{}_{|P|}C_n} \tag{7}$$

[6]スマートコントラクトからアクセス可能なランダムデータを r_{32} としておけば，送信が不要であるため効率的である．

共謀可能確率は $|P|$ を固定値とした場合 k と n および $|P\beta|$ に強い影響を受ける．式 (7) の $|P|$ を 100 で固定し，前述した 3 つの値をそれぞれ変動させた結果が，図 5 である．x 軸は共謀プロバイダの割合，y 軸は共謀可能確率となっている．これらの結果から，機密性においては $n = 5, k = 4$ の組み合わせが最も優秀であった．次いで，$n = 4, k = 3$ と $n = 6, k = 4$ の組み合わせが優秀であった．次点で優秀であったこの 2 つのパラメータ設定において，前者は手数料の面で，後者は可用性の面で優秀であろう．

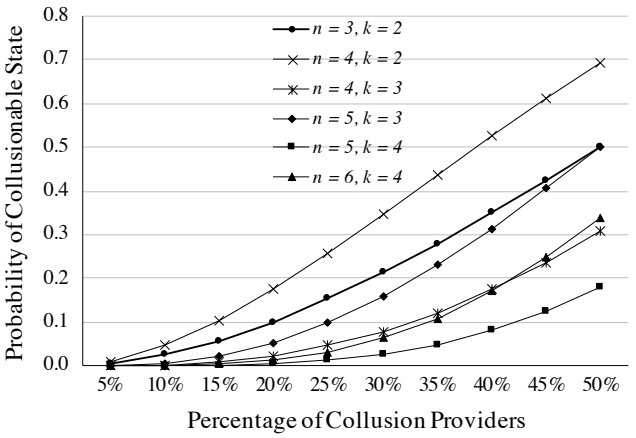

図 5: k, n および共謀割合と共謀可能確率

最後に，なりすまし攻撃を考える．p_i は c の本人認証を経て d_i を提供する．この本人認証において，プロバイダ毎にパスワードを設定して利用すると複数パスワードを記憶する負担がかかる．そこで，この負担を回避するために，Web サービスの認証を連携する仕組みである OAuth[7] や Auth0[8] を利用することが考えられる．しかし，これらの仕組みを利用するときに，単一のアカウントで全プロバイダの認証を行うと，プロバイダを分散した意味が薄れるため危険である．したがって，認証の連携を行う際，クライアントはプロバイダ毎に異なるアカウントで認証することが推奨される．

4.2. 可用性

c は pw を覚えていることと，p のオンラインサービスを利用できる状態にあることが前提条件となる．この前提条件のもとで，c が k 以上の p_i とリレーションを継続している場合は秘密鍵の復元が可能であり，これを満たさない場合は復元ができない．n や k を冗長化させると上記の復元条件を満たしやすくはなるものの，機密性の観点からは推奨できない．つまり，機密性にも考慮しながら高い可用性を維持するためには，リレーションを継続することが重要である．リレーションの継続率を高めるためには，c と p_i 双方にとって，リレーションの継続がナッシュ均衡[9] となる必要がある．ナッシュ均衡を作り出す方法の一つの例として，c と p_i の間での手数料を適切な価格に設定することが挙げられる．

5. 結論

本稿では，ブロックチェーン秘密鍵バックアップにおいての利便性・可用性の向上を目的として，秘密鍵のバックアップエコシステムを提案した．また，機密性・可用性の視点から考察を行った．機密性の考察では，KDF や SSS の適切なパラメータ設定が攻撃への対抗策となることが明らかになった．可用性の考察では，リレーションの継続が重要であることが明らかになった．今度の課題としては，より機密性を高めるために費用対効果とクライアントの本人認証について考察することと，より可用性を高めるためにリレーション継続のナッシュ均衡について考察することがあげられる．

参考文献

[1] FENG XIONG, RUIYANG XIAO, WEI REN, RONGYUE ZHENG, AND JIANLIN JIANG, "*A Key Protection Scheme Based on Secret Sharing for Blockchain-Based Construction Supply Chain System* " 2019 IEEE Access VOLUME 7, Pages 126773-126786, Aug-2019

[2] 山本博資, "秘密分散法とそのバリエーション（符号と暗号の代数的 数理）", 数理解析研究所講究録（2004）, 1361: 19-31, Apr-2004

[3] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich, "*Argon2: the memory-hard function for password hashing and other applications* ", PHC, March-2017

[7] OAuth 2.0 `https://oauth.net`

[8] Auth0 `https://auth0.com`

[9] ゲーム理論における非協力ゲームの解の一種．他のプレーヤーの戦略を所与とした場合，どのプレーヤーも自分の戦略を変更することによってより高い利得を得ることができない戦略の組み合わせ．どのプレーヤーも戦略を変更する誘因を持たない．