DEIM Forum 2022 J31-2

暗号化したブロックチェーン上データの解析における オフチェーンDB改ざん検知

Kim Byunghak† Le Hieu Hanh† 横田 治夫†

† 東京工業大学情報理工学院情報工学系 〒 152–8550 東京都目黒区大岡山 2-12-1
E-mail: †{kim,hanhlh}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

あらまし 近年、医療現場においてデジタル化が進み、電子カルテの導入が増えている。これにより医療情報解析による活用も更に活発になることが期待されている一方、共有時にプライバシーやセキュリティ、改ざんに関する問題が懸念されている。これまで、電子カルテの改ざん防止のためブロックチェーンを用い、プライバシー保護のためにプロキシ再暗号化を用いるアプローチが提案されているが、解析を効率良く行うためにはオフチェーン DB を用いることが望ましい。しかし、暗号化したブロックチェーン上データの解析におけるオフチェーン DB の改ざん検知は検討されていない。本研究では、暗号化されたデータをブロックチェーンに格納し、解析のためのオフチェーン DB の改ざん検知を行う場合の、複数のアプローチを検討し、Hyperledger を用いて実装したプロトタイプで評価を行う。

キーワード ブロックチェーン，プロキシ再暗号化，電子カルテ， オフチェーン DB，Hyperledger，改ざん検知

1 はじめに

1.1 研究背景

近年、社会の様々な分野にわたってデジタル化が進んでいる。資料をデジタル環境に保存した電子記録は、データの削除、変更、追加が簡単にできるため、事務効率の向上に効果があるといわれる。しかしデジタル化はデータ改ざんと情報漏洩のリスクが上がるという問題がある。このため、現在のデジタル化において機密性、完全性、可用性を維持することが重要な課題になっている [1]。

データ改ざん防止の方法の一つとしては、ブロックチェーンの導入がある。暗号通貨の基盤技術として開発されたブロックチェーンは、全ての取引が記録され、全ての台帳が共有されることでデータ不正に強い耐久性を持っている。

しかし、ブロックチェーンは遅い処理速度という問題を持っている。萱原の研究 [2] ではブロックチェーン上 (Hyperledger Fabric) と通常のデータベース上 (PostgreSQL) で 1 から 1000 までの総和計算を行い、その所要時間が 6800 倍の差が生じることを確認した。この点から、ブロックチェーン上でデータの解析を行うことは効率的でないことが分かる。

情報漏洩の対策の一つとしては、データの暗号化がある。情報漏洩は一度でも発生してしまうと、情報管理の主体は信頼を大きく失う。そのため、一度でも発生しないように防止しなければならないが、管理ミスや不正アクセスなどのすべての原因を完璧に防ぐことは現実的に難しいである。しかしデータが暗号化されていれば、情報流出が起きてもデータ解析を防ぐことが可能であり、そのデータの解析はデータの復号に必要な秘密鍵を持っている者に限ることも可能になる。

以上を踏まえると、ブロックチェーンを単独で使うよりブロックチェーンとオフチェーンを併用し、それぞれの長所を活用することが考えられる。データ解析を行う際にオフチェーンを使用すれば、より良いパフォーマンスが期待できる一方、データが改ざれる可能性がある。この恐れをブロックチェーンを活用して解消できれば、データの真正性も確保できる。またブロックチェーン上のデータを暗号化すれば、ブロックチェーンからの情報漏洩の防止も期待できる。

1.2 既存研究

1.2.1 ブロックチェーンとプロキシ再暗号化を用いた共有範囲設定可能な医療情報管理

萱原ら [3] の 2019 年の研究では医療情報の管理のため、Hyperledger を用いてデータをブロックチェーン上に記録し、プロキシ再暗号化を用いてユーザのアクセスレベルを設定できるアプローチを提案した。データはブロックチェーン上に暗号化されており、ユーザは自分のアクセスレベルが許可するデータだけ復号することが可能である。しかし、この研究の課題点としてオフチェーンでのデータ解析を考慮していない点があげられる。研究背景に述べたように、ブロックチェーンは速度の問題があり、データをオフチェーンで解析することでこのような問題を解決することが考えられるが、このようなオフチェーンへの拡張はここでは行われていない。

1.2.2 ブロックチェーン上データ解析用オフチェーン DB の改ざん検知と解析結果保証

萱原ら [2] の 2021 年の研究では、ハッシュ値比較を通じてオフチェーン DB のデータ改ざんを検知する手法を提案した。萱原らは、ブロックチェーンからデータを抽出したオフチェーン DB を解析する際に、データが改ざんされていないことを確認するために、ブロックチェーンのデータとオフチェーン DB のデータのハッシュ値を求めて比較した。比較後にデータが一致しない場合には、ブロックチェーンからデータを再抽出し、オフチェーン DB が正しいデータで維持されることを保証する。

再抽出する前にデータの Key とカラム情報にハッシュ値を求めて比較し、このハッシュ値が一致する場合、オフチェーン DB のデータがブロックチェーンのデータを正しく参照することが可能で、改ざんされたデータだけ再抽出するが、一致しない場合は Key とカラム情報まで改ざんがあったものと判断され、全データを再抽出する。しかし、この研究の問題点として、データの共有範囲設定とユーザの管理ができていない点が挙げられる。ブロックチェーンに参加する全てのユーザが全体データを見ることができるため、個人情報のように外部に流出してはならないデータを扱うことができなくなり、ブロックチェーンに新たな参加者を追加することも難しくなる。この研究で使われたブロックチェーンである Hyperledger Fabric では、プライバシーのようなセキュリティ問題を解決するために、ネットワークの特定の Peer が独立して参加できる channel や暗号化といった解決策を提供するが、ここでは導入されていない。

1.3 研究目的

本研究は暗号化したブロックチェーン上データの解析におけるオフチェーン DB 改ざん検知を目指す。その実現のため、暗号化したブロックチェーン上データとオフチェーン DB 上のデータの比較を行う 4 種類の改ざん検知手法を提案する。さらに、それぞれの手法のデータの格納・抽出・更新及びデータ改ざん検知の実行時間を比較する。

1.4 本稿の構成

本稿は以下の通り構成される。2 節では背景知識について説明する。3 節では、ブロックチェーン上の暗号文とオフチェーン DB 上のデータの比較法を提案する。4 節では 3 節の提案手法を実装し、結果の評価を行う。最後に 5 節では本研究のまとめと今後の課題を述べる。

2 背景知識

2.1 ブロックチェーン

ブロックチェーンは Satoshi Nakamoto が 2008 年に発表した論文 [4] で提案された仮想通貨であるビットコインの基盤となる技術のことを指す。日本ブロックチェーン協会は広義のブロックチェーンを「電子署名とハッシュポインタを使用し改竄検出が容易なデータ構造を持ち、且つ、当該データをネットワーク上に分散する多数のノードに保持させることで、高可用性及びデータ同一性等を実現する技術」[5] と定義している。

ブロックチェーンはその目的や運用方法によって様々に存在し、参加の制限有無でパブリック型とプライベート型で分類することが可能である。パブリック型ブロックチェーンは参加の制限がなく、だれでも自由に利用することが可能である。パブリック型ブロックチェーンはネットワークの維持のためノードを運用する参加者に仮想通貨を支給することが多い。パブリック型ブロックチェーンの代表例として、Bitcoin Core [6] や Ethereum [7] が挙げられる。プライベート型ブロックチェーンは参加の制限があり、管理者の許可を受けた参加者のみ利用が可能である。プライベート型ブロックチェーンはビズネス場面で使用される場合が多い。プライベート型ブロックチェーンで、多数の管理者が存在する場合があり、これをコンソーシアム型と呼ぶ。プライベート型およびコンソーシアム型の代表例として、Hyperledger が挙げられる。

ブロックチェーンの代表的な技術要素として「分散台帳」、「電子署名」、「コンセンサスアルゴリズム」、「スマートコントラクト」が挙げられる。分散台帳と電子署名、コンセンサスアルゴリズムによってブロックチェーンの参加者は取引データを共有し、信頼の中で取引を続けることが可能である。スマートコントラクトはブロックチェーン上で実行されるプログラムのことを意味する。ここでプログラムは契約であり、スマートコントラクトは契約条件が満たされた時、自動的に契約内容を実行する。この点から、ユーザは契約の締結まで待つことなく、予想可能な結果を得ることが可能である。スマートコントラクトの作成は主に高水準言語で行われており、Ethereum では Solidity と呼ばれる独自のプログラミング言語を使用する。

2.2 Hyperledger

Hyperledger [8] は、Linux Foundation が主管するブロックチェーンのオープンソースのプロジェクトであり、業界を超えた多くの企業が共同で参加する。Hyperledger は以下のような特徴を持つ。

- プライベート型ブロックチェーンとして、企業ビズネスを実現することに適している。
- 金融に特化した他のプラットフォームと異なり、様々な産業に汎用的に導入可能な技術標準を提示する。

Hyperledger プロジェクトの 1 つに Hyperledger Fabric [9] がある。Hyperledger Fabric はブロックチェーンソリューションと応用プログラムを開発するためのフレームワークである。Hyperledger Fabric は以下のような特徴を持つ。

- スマートコントラクトに一般のプログラミング言語の使用が可能である。
- 軽度で迅速なコンセンサスアルゴリズムを導入しており、コンセンサスアルゴリズムの交替が可能である。
- 台帳はチャンネルに分かれていて、チャンネルにアクセス可能なユーザを制限することができる。

また、Hyperledger Fabric を用いたアプリケーションを容易に構築できるようにする Hyperledger Composer [10] と呼ばれる開発ツールが用意されている。Hyperledger Composer はモデルファイルにビズネスの構成要素を定義し、スクリプトファイルでトランザクションを定義することでチェーンコードを作成可能であるが、2019 年 8 月を最後にサポートが終わり、追加的な開発は行われていない。しかし、本研究ではブロックチェーンのネットワークを簡単に構築できる点から提案手法の実装に Hyperledger Composer を用いた。

2.3 プロキシ再暗号化

再暗号化とは、ある公開鍵で暗号化された暗号文を、別の公開鍵で暗号化された暗号文に変換することである。また、再暗号化が第三者であるプロキシによって行われると、それはプロ

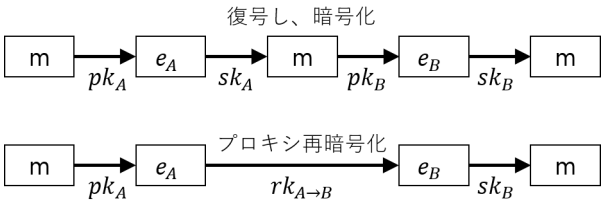

図 1　プロキシ再暗号化

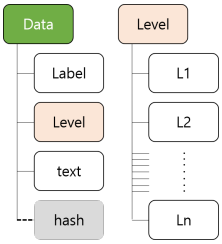

図 2　データ構造とアクセスレベル

キシ再暗号化と呼ばれる。

プロキシ再暗号化を用いると、第三者に秘密鍵を渡すことなく、再暗号化が可能になる。プロキシ再暗号化の流れを図 1 に示す。ユーザ A の公開鍵 pk_A で暗号化された暗号文 e_A をユーザ B の公開鍵 pk_B で暗号化された暗号文 e_B に変換する時、プロキシ再暗号化を用いない場合はユーザ A の秘密鍵 sk_A を使って復号し、ユーザ B の公開鍵 pk_B で暗号化する必要があるが、プロキシ再暗号化の場合は A から B への再暗号化鍵 $rk_{A\to B}$ を用いることでユーザ A 宛ての暗号文をユーザ B 宛ての暗号文に変換することができる。プロキシ再暗号化可能な暗号の 1 つには BBS 暗号 [11] がある。

3　提 案 手 法

本節ではデータを暗号化してブロックチェーン上に格納し、そのブロックチェーンからデータを抽出するモデルについて説明する。またブロックチェーン上の暗号化されたデータとオフチェーン上のデータを比較することで、データの改ざん検出を行う手法を提案する。

3.1　データ構造とアクセスレベル

本研究で、ブロックチェーンに格納されるデータの構造とアクセスレベルを図 2 に示す。ブロックチェーンの中で、データは Label、Level、text の三つの要素で構成されており、データのハッシュ値を利用するブロックチェーンは hash も追加されて四つの要素で構成される。Label はデータの固有の番号である同時にデータにおける識別子である。Level はデータのアクセスレベルである。アクセスレベルは l1, l2, ..., ln の n タイプがあり、l1 が一番高く ln が一番低いアクセスレベルである。データのアクセスレベルより低いアクセスレベルのユーザはデータを自分の暗号文に変換する再暗号化鍵が存在しないため、暗号文を復号することができない。ユーザは自分のアクセスレベル以下の再暗号化鍵を持つ。text は暗号化されたデータである。hash は暗号化されていないデータのハッシュ値である。

ブロックチェーンにはユーザのデータである Participants とユーザの再暗号化鍵である Reenckey のデータも存在する。Participants はユーザの ID で識別され、ユーザの情報が記録される。Reenckey はユーザ登録の際に鍵生成局から作成され、ユーザがもつ再暗号化鍵を保存する。Reenckey はユーザの ID で識別される。

3.2　データの格納と抽出

本研究におけるデータの格納・抽出の流れを図 3 に示す。ブロックチェーン格納されるデータはそのアクセスレベルに応じて暗号化される。

3.2.1　格　　納

格納の手順を図 3 のユーザ A の流れを用いて説明する。

（1）　ユーザ A がブロックチェーンに記録したいデータとそのデータのアクセスレベルを入力する。

（2）　アプリケーションは pk_A を用いて、データを暗号化する。

（3）　アプリケーションはデータ追加のリクエストをブロックチェーンに送る。

（4）　スマートコントラクトは $rk_{A\to l2}$ を用いて暗号文を再暗号化し、ブロックチェーン上に記録する。

データのハッシュ値を利用するブロックチェーンはデータにハッシュ関数をかけ、データ追加のリクエストにハッシュ値を含むことでブロックチェーン上にハッシュ値を記録する。

3.2.2　抽　　出

抽出の手順を図 3 のユーザ B の流れを用いて説明する。

（1）　ユーザ B がブロックチェーンから読み取りたいデータの Label を入力する。

（2）　アプリケーションは Label とユーザの Id を用いて、データ復号のリクエストをブロックチェーンに送る。

（3）　スマートコントラクトはアクセスレベル $l2$ の再暗号化鍵 $rk_{l2\to B}$ を用いて暗号文を再暗号化する。

（4）　スマートコントラクトは再暗号化された暗号文をアプリケーションに送る。

（5）　アプリケーションは sk_B を用いて、復号を行う。

（6）　アプリケーションはデータをユーザに渡す。

また、図 3 でユーザ C はデータ抽出に失敗することを確認できる。ユーザ C はデータより低いアクセスレベルのユーザであり、データの再暗号化鍵が存在しない。そのため、アプリケーションに送られるデータは再暗号化されていないデータであり、ユーザ C の秘密鍵 sk_C では復号不可能である。

データのハッシュ値を利用するブロックチェーンでも、データ抽出はハッシュ値を利用しないブロックチェーンと同じ手順で行われる。

3.3　改ざん検知

本研究で提案する改ざん検知の手法の流れを図 4 に示す。暗号化されたデータと暗号化されていないデータを比較するためには、両者を比較可能なデータに変換する必要がある。本節で提案する手法は、比較可能なデータの形式を何にするかによっ

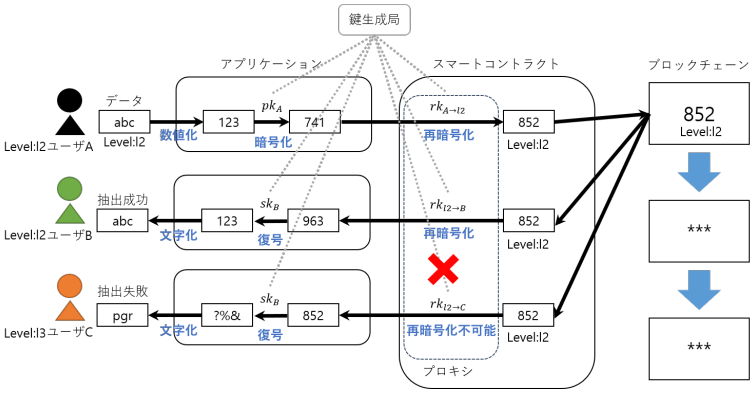

図 3　提案手法：格納と抽出

て分類される。

3.3.1　手法 1 平文比較

平文比較では、データを平文に変換して比較を行う。平文比較の手順を図 4 のユーザ D の手法 1 の流れを用いて説明する。

（1）　ユーザ D がブロックチェーンから改ざん検知したいデータを入力する。

（2）　アプリケーションはデータの Label とユーザの Id を用いて、データ復号のリクエストをブロックチェーンに送る。

（3）　スマートコントラクトはアクセスレベル $l2$ の再暗号化鍵 $rk_{l2\to D}$ を用いて暗号文を再暗号化する。

（4）　スマートコントラクトは再暗号化された暗号文をアプリケーションに送る。

（5）　アプリケーションは sk_D を用いて、復号を行う。

（6）　アプリケーションはユーザ D が入力した平文のデータとブロックチェーンからの平文のデータを比較する。

平文比較の手法で、ユーザが入力したデータは暗号化せず、ブロックチェーンで再暗号化して抽出した暗号文は平文に復号して両者を比較する。

3.3.2　手法 2 ユーザの暗号文比較

ユーザの暗号文比較では、データをユーザの公開鍵で暗号化された暗号文に変換して比較を行う。ユーザの暗号文比較の手順を図 4 のユーザ E の手法 2 の流れを用いて説明する。

（1）　ユーザ E がブロックチェーンから改ざん検知したいデータを入力する。

（2）　アプリケーションは pk_E を用いて、データを暗号化する。

（3）　アプリケーションはデータの Label とユーザの Id を用いて、データ復号のリクエストをブロックチェーンに送る。

（4）　スマートコントラクトはアクセスレベル $l2$ の再暗号化鍵 $rk_{l2\to E}$ を用いて暗号文を再暗号化する。

（5）　スマートコントラクトは再暗号化された暗号文をアプリケーションに送る。

（6）　アプリケーションはユーザ E が入力したデータの暗号文とブロックチェーンからの暗号文を比較する。

ユーザの暗号文比較の手法で、ユーザが入力したデータはユーザの公開鍵で暗号化し、ブロックチェーンで再暗号化して抽出した暗号文と比較する。

3.3.3　手法 3 ブロックチェーンの暗号文比較

ブロックチェーンの暗号文比較では、データをブロックチェーンの公開鍵で暗号化された暗号文に変換して比較を行う。ブロックチェーンの暗号文比較の手順を図 4 のユーザ F の手法 3 の流れを用いて説明する。

（1）　ユーザ F がブロックチェーンから改ざん検知したいデータを入力する。

（2）　アプリケーションはデータの Label とユーザの Id を用いて、データ抽出のリクエストをブロックチェーンに送る。

（3）　スマートコントラクトは暗号文をアプリケーションに送る。

（4）　アプリケーションは pk_l2 を用いて、データを暗号化する。

（5）　アプリケーションはユーザ F が入力したデータの暗号文とブロックチェーンからの暗号文を比較する。

ブロックチェーンの暗号文比較の手法で、ユーザが入力したデータはブロックチェーンの公開鍵で暗号化し、ブロックチェーンから抽出した暗号文と比較する。

3.3.4　手法 4 ハッシュ値比較

ハッシュ値比較では、データをハッシュ値に変換して比較を行う。ハッシュ値比較の手順を図 4 のユーザ G の手法 4 の流れを用いて説明する。

（1）　ユーザ G がブロックチェーンから改ざん検知したいデータを入力する。

（2）　アプリケーションはデータにハッシュ関数を適用する。

（3）　アプリケーションはデータの Label とユーザの Id を用いて、データ抽出のリクエストをブロックチェーンに送る。

（4）　スマートコントラクトはハッシュ値をアプリケーションに送る。

（5）　アプリケーションはユーザ G が入力したデータのハッシュ値とブロックチェーンからのハッシュ値を比較する。

ハッシュ値比較の手法で、ユーザが入力したデータにハッシュ関数を適用し、ブロックチェーンに記録されていたハッシュ値を抽出して両者を比較する。

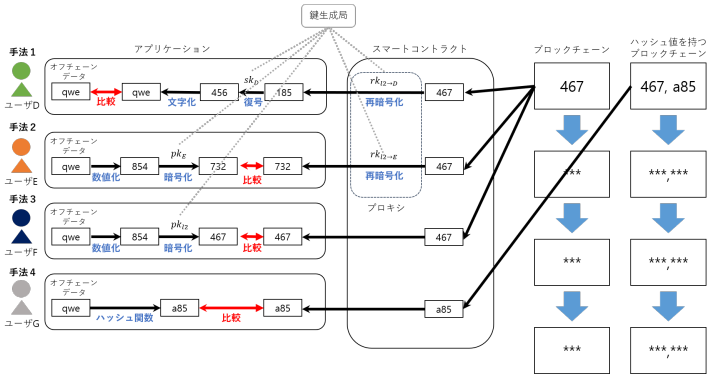

図 4　提案手法：改ざん検知

3.4　更　　新

改ざん検知の提案手法をみると手法 4 はブロックチェーンにデータのハッシュ値を持つ。この性質によって、手法 4 は手法 1,2,3 とは違うブロックチェーンを必要とし、格納の時にハッシュ値も記録する必要がある。ハッシュ値を導入することによって、手法 4 の改ざん検知の流れは手法 1,2,3 に比べると簡単になった。

しかしデータのハッシュ値を暗号化されたデータと記録するとデータを更新したい場合、データのハッシュ値を再計算するためデータの復号が必要になる。この点を確認するため、本節ではデータの更新手法を提案する。更新は複数のデータから多数の値を一度に変えることも可能であるが、本研究の手法では一つのデータから一つの値を更新することとする。データの中で更新したい部分は、コラム番号を通して伝えることにする。

3.4.1　手法 1,2,3 ハッシュ値を持たないブロックチェーン

ハッシュ値を持たないブロックチェーンの更新の手順を図 5 ユーザ A が更新する場合を例として説明する。この図ではデータにかかる計算の流れとそれによるデータの変換を表記した。

（1）　ユーザ A がブロックチェーンから更新したいデータとカラム番号を入力する。

（2）　アプリケーションはデータを暗号化する。

（3）　アプリケーションはブロックチェーンにデータ更新のリクエストをブロックチェーンに送る。

（4）　スマートコントラクトはデータを再暗号化し、ブロックチェーンの暗号文のカラム番号の部分を入れ替える。

3.4.2　手法 4 ハッシュ値をもつブロックチェーン

ハッシュ値をもつブロックチェーンの更新の手順を図 5 ユーザ B が更新をする場合を例として説明する。この図ではデータにかかる計算の流れとそれによるデータの変換を表記した。

（1）　ユーザ A がブロックチェーンから更新したいデータとカラム番号を入力する。

（2）　アプリケーションは Label とユーザの Id を用いて、データ復号のリクエストをブロックチェーンに送る。

（3）　スマートコントラクトは暗号文を再暗号化する。

（4）　スマートコントラクトは再暗号化された暗号文をアプリケーションに送る。

（5）　アプリケーションは sk_B を用いて、復号を行う。

（6）　アプリケーションはデータのカラム番号の部分を新しいデータに入れ替える。

（7）　アプリケーションは新しくできたデータにハッシュ関数を適用する。

（8）　アプリケーションは pk_B を用いて、データを暗号化する。

（9）　アプリケーションはブロックチェーンにデータ更新のリクエストをブロックチェーンに送る。

（10）　スマートコントラクトはてデータを再暗号化し、ブロックチェーンの暗号文のカラム番号の部分を入れ替える。

（11）　スマートコントラクトはハッシュ値を入れ替える。

図 5 は手法 1，2，3 と手法 4 におけるデータ更新の例を示す。元データの 123,456,789 を 963,456,789 に更新すると考える。ハッシュ値を持たないブロックチェーンの手法 1，2，3 では更新する部分の 123 から (963) に変更し、暗号化（741）と再暗号化（812）を行った後に、812 をブロックチェーンに格納する。一方、ハッシュ値を持つブロックチェーンの手法 4 では、まず全体のデータを抽出し、部分的に更新した後に、全体を暗号化した値（715,138,658）をブロックチェーンに格納する。それと同時に、更新したデータ（963,456,789）のハッシュ値（ht8）もブロックチェーンに格納する必要がある。

4　実　　験

本節では提案手法で説明した格納、抽出、改ざん検知、更新の所要時間に関する評価実験を行う。実験で動作確認を行ったところ、アクセスレベルに応じて暗号化および復号、改ざん検知が適切に行われていることが確認できた。

本実験で使われるブロックチェーンのネットワークは Hyperledger Composer を用いて作成した。本実験で使われるユーザのアプリケーションは Node.js によって記述されており、ブロックチェーンのスマートコントラクトは Javascript によって記述されている。本実験ではハッシュ関数として SHA-256 を採用した。本実験では暗号方式として BBS 暗号を使用した。

本実験で使用された暗号化方式は改ざん検知の手法によって異なる。暗号文で比較を行う手法 2,3 は比較を行うためには同じ平文に対して常に同じ暗号文が変換される必要がある。そのため手法 2,3 の改ざん検知は常に同一の暗号文に変換される決

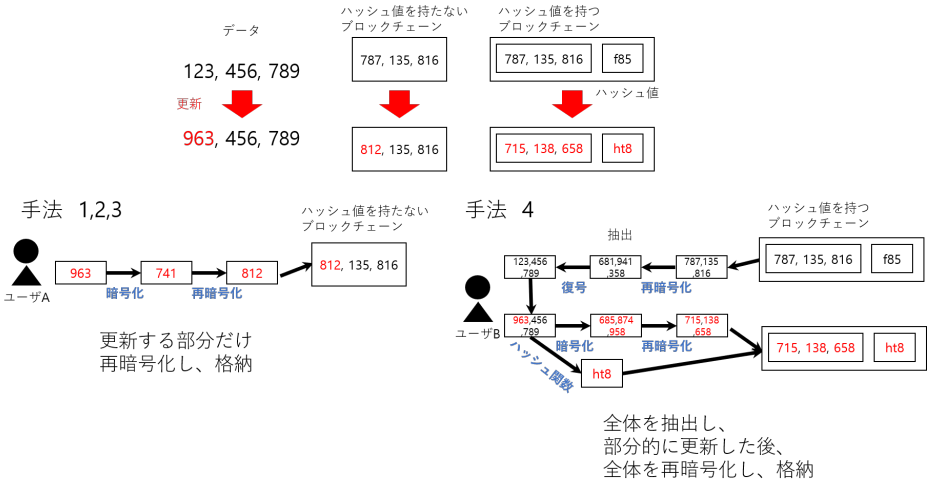

図 5　提案手法：更新

表 1　実 験 環 境

CPU	Intel Core i5-8250U 1.6GHz
Memory	DDR4 8GB 2400MHz
OS	Ubuntu 18.04.6 LTS
Docker	version 20.10.11, build dea9396
Hyperledger Fabric	v1.2.1
Hyperledger Composer	v0.20.9
Hyperledger Composer Playground	v0.20.9
Hyperledger Composer REST Server	v0.20.9

定的暗号で暗号化を使用した。しかし、平文とハッシュ値で比較を行う手法 1,4 は比較を行うために常に同じ暗号文が変換される必要がない。そのため手法 1,4 の改ざん検知は暗号文は同じ平文に対しても異なる暗号文に変換される確率的暗号で暗号化を行った。

実験環境は表 1 に本実験の実験環境を示す。

4.1　実 験 内 容

本実験ではデータの格納と抽出・更新を行う。データの格納と抽出・更新は三つの条件があり、それは次のようである。

- A：ハッシュ値を持たないブロックチェーンの確率的暗号化
- B：ハッシュ値を持たないブロックチェーンの確定的暗号化
- C：ハッシュ値を持つブロックチェーンの確率的暗号化

その後、後述する改ざんパターンにおいて改ざん検知の提案手法 4 つの改ざん検知を行う。格納、抽出、改ざん検知、更新は各条件と手法において 5 回ずつ行い、所要時間を計測しその平均を求める。

本実験のデータの単位はタプルであり、データの格納と抽出、改ざん検知はタプル単位で行われる。本実験ではタプル数 100、カラム数 100 の xlsx ファイルをデータとして扱う。データ中の一つの項目は 8 バイトのランダムな文字列になっている。

なおオフチェーンのデータには 2 パターンの改ざんがある。

- All MissMatch：全てのタプルにおいて改ざんが検知される。
- All Match：データの改ざんは検知されない。

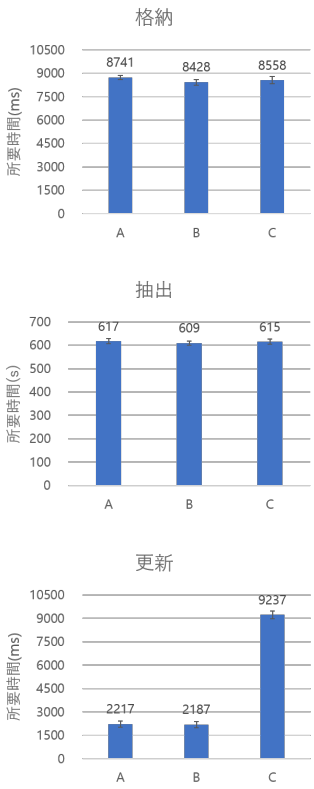

図 6　(a) 格納 (b) 抽出 (c) 更新

4.2　実 験 結 果

ハッシュ値を持たないブロックチェーンとハッシュ値をもつブロックチェーンの格納、抽出、更新の所要時間の平均を図 6 の (a),(b),(c) に示す。改ざん検知手法の手法 1 平文比較、手法 2 ユーザの暗号文比較、手法 3 ブロックチェーンの暗号文比較、手法 4 ハッシュ値をもつブロックチェーンを図 7 に示す。

4.3　考　　察

データの格納と抽出と更新から、確率的暗号と確定的暗号の

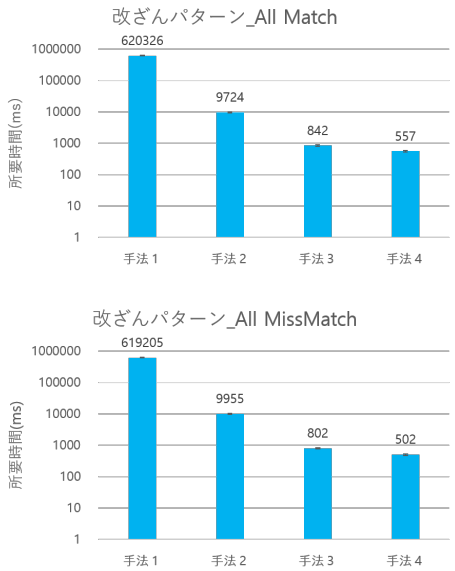

図 7　(a) 改ざん検知 (All Missmatch) (b) 改ざん検知 (All Match)

所要時間の差は大きくなかった。本実験で確定的暗号は確率的暗号のランダムな数字を固定することであったが、この方式では暗号化と復号、再暗号化の所要時間に大きな影響を与えないことが分かった。

データの格納はすべての条件で 10 秒もかからなかったが、データの抽出はすべての条件で 10 分以上かかった。格納はデータの暗号化と再暗号化で構成されており、抽出は暗号文の再暗号化と復号で構成されていることから復号が暗号化よりも時間を要することが分かる。これは萱原 [3] の 2019 年の研究でも確認できる。

データの更新はハッシュ値を持たないブロックチェーンがハッシュ値を持つブロックチェーンより短い時間で作業を終えた。ハッシュ値を持たないブロックチェーンはデータの暗号化と再暗号化で構成されているが、ハッシュ値を持つブロックチェーンはデータの再暗号化と復号と暗号化と再暗号化で構成されているためだと考えられる。

オフチェーンデータの改ざんパターンからも所要時間の大きな差は見られなかった。全てのタプルを比較することで、動作に違いがなく、本実験では改ざん検知以降に追加作業がなかったためであると考えられる。

改ざん検知で手法 1 は抽出とほぼ同じ所要時間を要した。手法 1 と抽出はデータをブロックチェーンから抽出し、それをファイルに保存するか入力されたデータと比較するかの違いがあったが、この作業は復号に比べると所要時間が短ったことに考えられる。また手法 3 より手法 4 の所要時間が 300ms ほど短いことから、データの暗号化よりハッシュ関数の適用が早く終わることが分かる。

5　おわりに

5.1　まとめ

本研究では、様々なアクセスレベルのユーザがデータを暗号化してブロックチェーン上に格納して抽出するシステムを提案し、オフチェーン DB 上のデータが改ざんされていないか検証できる改ざん検知の手法を提案した。

さらに、提案方法のプロトタイプを実装し、データの格納・抽出・更新及び改ざん検知の所要時間の評価実験を行った。その結果、データの格納は BBS 暗号化方式において確率的であれ決定的であれ所要時間に大きく影響せず、データの抽出は復号が多く行われるため、データの格納と比べて 70 倍以上の時間がかかることが分かった。データの改ざん検知は、ブロックチェーンにあらかじめ検証用データであるハッシュ値を格納しておくことで、時間のかかる復号や再暗号化を行わないことが可能になり、所要時間が最大 1100 倍以上減少することが確認できた。しかし、その場合データを更新する際にはハッシュ値を更新するためにデータを復号する過程が生じるため、ハッシュ値を持つブロックチェーンがハッシュ値を持たないブロックチェーンより 4 倍以上所要時間が長かった。

5.2　今後の課題

今後のの課題としては、まず本研究の実験をさらに補完し追加的実験を行うことである。今回の実験で改ざん検知の比較単位はタプルに固定したが、比較はタプル以外にもテーブル、カラム、アイテムの単位でも行うことが可能である。改ざんのパターンは 2 つであったが、改ざんされたデータの量によって、いくつかのパターンを追加することも考えられる。また本研究の実験データより大容量のデータで実験を進めることも挙げられる。本研究の実験で使用したデータのタプル数は 100 であったが、タプル数を 1000 とすると Hyperledger Fabric のタイムアウトにより実験を進めることができなかった。ネットワークのタイムアウト値をさらに大きくすることで、実験に使用されるデータの運用幅を広げることが期待できる。

文　献

[1] JISQ27001, 情報技術－セキュリティ技術－情報セキュリティマネジメントシステム－要求事項, 日本産業標準調査会, 2014/03/20
[2] 萱原正彬, Hieu Hanh Le, 横田治夫. ブロックチェーン上データ解析用オフチェーン DB の改ざん検知と解析結果保証. 第 13 回データ工学と情報マネジメントに関するフォーラム, 2021.
[3] 萱原正彬, 本田祐一, 山田達大, Hieu Hanh Le, 串間宗夫, 小川泰右, 松尾亮輔, 山崎友義, 荒木賢二, 横田治夫. ブロックチェーンとプロキシ再暗号化を用いた共有範囲設定可能な医療情報管理. 第 11 回データ工学と情報マネジメントに関するフォーラム, 2019.
[4] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
[5] ブロックチェーンの定義, http://jba-web.jp/archives/2011003blockchain_definition
[6] Bitcoincore, https://bitcoincore.org/
[7] Ethereum, https://ethereum.org/en/
[8] Hyperledger, https://www.hyperledger.org/
[9] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian

Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. 2018

[10] Hyperledger Composer, `https://hyperledger.github.io/composer/latest/`

[11] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In the proceeding of the International Conference on the Theory and Applications of Cryptographic Techniques, pp. 127–144. Springer, 1998.