ブロックチェーンと秘密分散法を用いた情報ライフサイクル制御

今田 丈雅[1]　松浦 幹太[1]

概要：われわれは今日なんらかのコミュニケーションツールを使って情報の伝達をしている．情報伝達にあたっては中央サーバーを経由しており, ユーザーからはデータの制御権が離れてしまう．このため後から政府の検閲によってセンシティブなデータが見られてしまう可能性がある．そのような脅威に対抗するためにはデータに期限を設定し, 自動で消去されるような仕組みが必要である．
本研究では公開分散型台帳と秘密分散法を組み合わせることによって上記の要請を満たす仕組みを提案する．このシステムは従来研究と異なり, 信頼できる第三者機関やセキュアなハードウエアを必要とせずシビル攻撃にも耐性があるという性質を持つ．

キーワード：ブロックチェーン，秘密分散法，消去，検閲

Information Lifecycle Control Employing Blockchain and Secret Sharing

Takemasa Imada[1]　Kanta Matsuura[1]

Abstract: Today we are communicating information using some kind of communication tool. In communication, information goes through the central server and users lose its control. Therefore, sensitive data might be revealed by government censorship in the future. In order to counter such threats, it is necessary for data to set an expiration time and to be automatically deleted.
In this research, we propose a mechanism that satisfies the above requirement by combining public blockchain and secret sharing. This system differs from previous research in that it is not requiring trusted third parties or secure hardware and resistant to a Sybil attack.

Keywords: blockchain, secret sharing, deletion, censorship

1. 序論

1.1 背景

通信の秘密 [1] は私生活の保護ないしプライバシーの保護の観点から守られるべきである．一方で, 重大な犯罪に関係する被疑者が罪を犯したと疑うに足る十分な理由がある場合, 公権力による通信の傍受は許されるとされている [2]. デジタルな世界における通信の検閲の具体例としてはメールやメッセージの検閲があるが, 実際にメールサービスが公権力の要求にしたがってメールの平文を提供していることが知られている [3,4]. しかし, 公権力の乱用によって傍受される対象でないものまで傍受されていることがスノーデン事件によって明らかになった [5]. したがって, われわれはデジタル通信への公権力の乱用に対する自衛策を持つ必要がある．

ここで, デジタルな世界におけるコミュニケーションの性質について考えると, われわれは普段情報の伝達を行う時になんらかのコミュニケーションツールを介しているということがわかる．つまり, コミュニケーションを行う場合にあたって, 情報は企業の持つ中央サーバー（クラウ

[1] 東京大学生産技術研究所
Institute of Industrial Science, The University of Tokyo

© 2017 Information Processing Society of Japan

ド）を経由するので情報はわれわれの手から離れ制御を失う．したがってコミュニケーションによって伝達された情報はユーザーが意図しない形で中央サーバーに永久に残る可能性がある．例えば, Facebook では一度画像をアップロードしたら, その画像を削除した場合でも数カ月は削除されたはずの画像にアクセスできたという事例が報告されている [6]．また, iCloud では削除されていたはずの芸能人のプライベートな画像が流出した事例が存在する [7]．これらの事例が発生した根本的な原因として, 一つはクラウドサービスではデータの可用性を向上させるためにスナップショットによるクラウド内データのバックアップがとられており, 根本的にデータを完全消去するのが不可能な設計になっているということがある．二つ目は, サーバーやクラウドは管理者しか中の構造を見ることができないブラックボックスであるために本当にサーバーから消去されているかは一般の利用者には知りようがないということがある．したがってクライアントやブラウザーからコンテンツを削除することができたとしても実はサーバーやクラウドには消したはずのデータが残っている可能性があり, 誰もが検証できる形でクラウドから削除することは難しい．

メールデータにおいても Facebook や iCloud の事例のようにサーバーのバックアップの仕組みによってデータがサーバーに残り続けている可能性は十分にあり, 機密性の高いメールをユーザー側がクライアントやブラウザー上のメニューから手動で削除してメールを一覧から削除したとしても実際には消えてはおらず, 忘れたころに公権力によってメールサービスに対して召喚状が発行されるなどしてメールの中身が見られてしまう可能性がある．

では, そもそもメールサービスにすらメールの内容がわからないようにすれば検閲を防ぐことができるのではないだろうか．これは End-To-End による暗号化メールサービスの利用が有力な候補として挙げられる．しかし, このやり方では検閲に対して不十分である．例えばイギリスにおいては鍵を用いてデータの暗号化を行っている人には鍵の提出を求めることができる法律があり [8], 公権力がメールの受信者の鍵を手に入れてメールを復号したり, 復号させたりできる．

このように, 検閲の力はデータのプライバシーを考える上で非常に脅威である．われわれは, メールサービスの持つすべてのデータやメールの受信者の持つ復号鍵はいつか漏えいすると考えて, その上でデータのプライバシーが守られるような仕組みを考える必要がある．

上記を背景として, retroactive privacy という安全性の性質に関する研究が存在する．retroactive privacy とはデータがある一定期間経過すると他人やひいてはメッセージのやり取りをしていた人でさえもデータにアクセスできなくなるという性質のことである．代表的な研究として, 検閲耐性を背景とした信頼できる第三者機関とセキュアなハードウエアを仮定しないデータの自動消去システムとして 2009 年に Vanish [9] が登場した．Vanish は分散ハッシュテーブル（DHT）と秘密分散法を組み合わせてデータの自動消去を実現している．しかし, 別の研究者によって DHT へのシビル攻撃を利用することで効率的な攻撃が存在することが示されている [10]．シビル攻撃を利用したある攻撃に対して耐性のある Vanish の方式が提案されてはいる [11] が, 依然としてシビル攻撃に対してぜい弱であるということが知られている [12]．

そこで本研究ではパブリックブロックチェーンの持つシビル攻撃への耐性と秘密分散法を利用した, データに期限時刻を設定し, 期限時刻以降になると自動的にデータが復号できなくなるプロトコルを提案する．さらに, プロトタイプ実装を行い実用性を計測する実験を行った．われわれの提案は強いセットアップの仮定を必要とせず, 鍵管理機関を信頼せずともよく, シビル攻撃にも強いという性質を持つ．本研究で提案する自動消去プロトコルの応用例は広く, メールのほかにも SMS や MMS などのメッセージサービス, SNS の個人間チャット, および個人的なファイルにも利用が可能である．

本研究ではメッセージのやり取りからしばらくして, 公権力の検閲によりメッセージの漏えいが起きる状況を対象としている．したがって, データにアクセスできる期間中に, 特定の人物がやり取りしていたメッセージに対する検閲からデータのプライバシーを守ることは本研究の対象外であることに注意されたい．

本研究の体裁は以下の通りである．2 節では本研究で達成する性質やシステムのモデルおよびプロトコルへの攻撃者モデルを定める．3 節では retroactive privacy を達成するために考えられる手法について検討を行う．4 節では retroactive privacy に関する既存の研究について紹介する．5 節では本研究で提案するプロトコルについて説明を行う．6 節では提案するプロトコルの安全性について定性的な分析を行う．7 節では提案するプロトコルについてプロトタイプ実装を行い, パフォーマンス評価を行った．最後に 8 節では本研究の貢献のまとめと, 研究の展望について述べる．

2. システムのモデルと目標

2.1 参加者

提案するプロトコルにおける参加者は送信者と受信者とブロックチェーンネットワークとする．送信者が受信者に対して, メールやメッセージサービスなどのコミュニケーション媒体を使って期限時刻以内ならば復元できるデータを送る状況を設定する．実際に送信者が受信者に送るものは平文ではなく, 平文をカプセル化したデータ（以降 VDO とする）である．

本研究における提案手法で用いるブロックチェーンネットワークは基本的には通常のブロックチェーンと似たも

© 2017 Information Processing Society of Japan

のである．ここではブロックチェーンの定義をブロックがチェーン状につながったデータベースおよびデータベースを維持していく仕組みの総称であるとし，今後ブロックチェーンとだけ記述があったらそれはパブリックブロックチェーンを意味するものとする．Bitcoin 型 [13] のブロックチェーンとの差異については提案手法の節で説明する．

2.2 目標

われわれの目標は指定された時刻まではデータにアクセスできるが，それ以降はあらゆる参加者からのデータアクセスを防ぐシステムを設計することである．

ここで，期限時刻である t_e は送信者側のクロックからみた値であることに留意されたい．われわれのシステムが達成すべき安全性やその他の性質は以下である．

- retroactive privacy
- 可用性
- 送信者および受信者は常時オンラインである必要はない
- セキュアなハードウエアや信頼できる第三者の機関を必要としない

retroactive privacy とは送信者によって期限時刻 t_e が設定されたメッセージ m について期限時刻以降はどの参加者からもアクセス不可になるという性質である．一方データの期限時刻 t_e 以前は，送受信者はデータにアクセスできる．

次に，可用性とは受信者がデータの生存期間中はデータの復号鍵のシェアを取得してデータの中身にアクセスできることを意味する．

また，今回提案するプロトコルではコミュニケーションが同期を必要とするものでもしないものでも機能するということである．つまり，送信者と受信者は同じ時間にオンラインである必要がなく，特にデータの生存期間の間はいつでもオフラインになることができる．

2.3 仮定

2.3.1 送信者と受信者間に安全な通信路がある

これはデータの期限切れになるまで VDO は漏えいしないということを意味する．送信者や受信者がメールサービスを利用する時のエンドユーザーとメールサービス間の通信や SMTP サーバーと POP サーバー間の通信で SSL や TLS といったプロトコルを使って認証されかつ暗号化された通信を行えばこの仮定は満たされる．

2.3.2 送信者と受信者は使用後の鍵を速やかに削除する

送信者と受信者は鍵をデータの暗号化や復号に使用した後に鍵が用済みになったら速やかに鍵を完全消去するという仮定である．

2.3.3 送信者は常に正しいチェーンを得ることができる（SPV 仮定）

ブロックチェーンにおいて正しいチェーンというのはネットワーク内に複数存在するチェーンのうち最長であることと，ブロックのフォーマットが正しいということと，ブロックのトランザクション間の関係が論理的に整合性がとれていることを満たしたチェーンのことである．

本研究で提案する手法において送信者は SPV クライアントとしての機能を有する．SPV クライアントとはフルノードのように全てのチェーンをダウンロードせず，ブロックのヘッダーのみをダウンロードするようなクライアントである．SPV クライアントは honest のハッシュパワーが常に攻撃者に勝っていることを仮定している．つまり，SPV クライアントは最長のチェーンの歴史が常に正しいということを暗に信頼していることになり，例えブロックの内容が無効なものであっても正しいチェーンとみなしてしまう．したがって送信者は常にブロックチェーンネットワークから正しいチェーンのヘッダーを受け取ることができるという仮定が必要になる．

2.3.4 マイナーは十分なストレージ，生存期間を持つ

マイナーは十分なストレージを持ち，ノードとしての生存時間もある程度長いことを見込むことができる．この仮定はシステムの可用性と密接にかかわる．マイナーは全チェーンをローカルに持っているのでシェアを保持するぐらいの余裕はあると考えられる．さらに，マイナーはマイニングを行うためにかなり長期間オンラインであることが考えられるのでそこまで強い仮定ではない．

2.3.5 弱同期モデル

送信者と受信者とブロックチェーンノードは弱同期していると仮定する．すなわち，送信者と他の参加者のデータの期限時刻時点におけるクロックの最大誤差が Δ を上回らないということである．言い換えれば，送信者のクロックが t_e を指した時にほかの参加者のクロックが $t_e - \Delta$ から $t_e - \Delta$ の間にあるということである．

2.4 攻撃者モデル

本システムが達成する retroactive privacy に対して 2 段階の攻撃者モデルを考える．攻撃者が用いるアルゴリズムは全て確率多項式時間アルゴリズムとする。

2.4.1 $t \leq t_e + \Delta$ における攻撃者モデル

攻撃者はネットワークに関して強大な力を持つ．例えば，ネットワークを盗聴したり，妨害したり，改ざんしたりすることが可能である．ただし，プロトコル参加者を汚染することはできないものとする．現実世界では汚染された ISP やメールプロバイダー，リレーノードなどはこのモデルに該当するといえる．

2.4.2 $t > t_e + \Delta$ における攻撃者モデル

期限時刻より前のモデル，つまりネットワークに対してかなり強力な力を持つのに加えて期限時刻以後ではプロトコルに参加しているあらゆる参加者を汚染できる．汚染できるというのは参加者の持つすべての情報を手に入れるこ

© 2017 Information Processing Society of Japan

とができるしプロトコルから逸脱したどのような動作も許されるということである.

3. 手法の検討

送信者と受信者が何らかのメッセージサービスを介してメッセージをやり取りしている状況において retroactive privacy が達成されるためにはデータを時間通りに, 安全に消すことを行わなければならない. ここではどのようにデータを消すのが retroactive privacy を満たす上で最適であるかということについて議論する.

3.1 完全消去

データの完全消去の手法としては上書きによる削除や鍵による消去が存在する [14]. 上書きによる削除は該当のメモリー領域そのものをランダムなデータで上書きするといったものであり, 鍵による消去 [15] はディスクに保存する前に暗号化を行い, 適切な時に復号鍵を消去することによってデータにアクセスできなくさせるというものである. データが信頼できないサーバー上に存在するという状況では鍵による消去が最も有効であると考えられる. なぜならば, 復号鍵を自分が管理していれば暗号化されたデータが存在しているサーバーが信頼できなくともデータを安全に消去できるからである. また, この手法ではデータをファイルシステムと外部のバックアップから同時に削除できる利点がある.

3.2 適時削除

次に, 鍵による消去の方法を用いたうえで時間通りにデータを消去することについて議論する. 考えられるのが送信者と受信者が期限時刻ちょうどに復号するための鍵を速やかに消すという方法である. しかしながら, この方法は常に時間通りにデータを消すことができるという要求を満たさない. 例えば受信者が期限時刻前後にシャットダウンもしくはクラッシュなどしてクライアント PC の電源が落ちていたりする場合には受信者は期限時刻ちょうどに復号鍵を消去することができず, 期限時刻以降にも復号鍵は残り続けることになる. つまり, 時間通りにデータを消すことを実現するためには送信者と受信者のみで完結した仕組みを作ることは難しいと考えられる.

では, 復号鍵を管理する第三者を設定した場合ではどうであろうか. この場合は第三者が期限時刻に復号鍵を速やかに消すという仮定を置けば, 受信者は復号するタイミングにのみ第三者から復号鍵を受け取ればよい. したがって, 長時間復号鍵を保持することなく期限時刻より前はいつでもデータにアクセスすることが可能になるし, 逆に期限時刻後はデータを復号できないことがある程度保証される.

3.3 まとめ

以上の議論から retroactive privacy を満たすためには鍵による消去を用いて, 鍵は第三者の鍵管理機関に預けるという方式が適当であるという結論が導かれた. 実際に retroactive privacy に関する研究ではすべて第三者に復号鍵を預ける方法がとられている.

4. 関連研究

retroactive privacy を実現する手法の初出としては Ephemerizer [16] がある. Ephemerizer では保護されたコンテンツにアクセスするために信頼できる第三者の鍵管理機関が必要となる. 鍵管理機関はユーザーに指定された時間が経過したら自動的にコンテンツを削除する. 3 節で述べたように retroactive privacy の性質を満たすなら Ephemerizer のように信頼できる第三者機関を設定してしまえば十分である. しかし, 鍵管理機関に信頼を置くことは検閲耐性のあるシステムとしては不適である. したがって, 単一の鍵管理機関の汚染に対して頑強であるためには鍵管理機関は分散されている必要がある.

分散された鍵管理機関の管理方法としては Tor [17] のディレクトリサーバーのように何個かのサーバー間で鍵管理機関の共通リストを持っておくことが考えられる. Tor のディレクトリサーバーは全部で九つ存在するがそのうち五つ以上が汚染されていなければ整合性のあるリレーノードのリストを持ち続けることができる仕組みになっている [18]. しかし, アメリカ国家安全保障局（NSA）が Tor のディレクトリサーバーのうち二つを汚染する計画を立てていたことが明らかになったこともあり [18], Tor で行われているリストの管理方法も安全とは言い難い. したがって, 鍵管理機関は分散されておりかつ, 鍵管理機関のリストを管理する主体が存在しないことが望ましい.

以上の要請を満たしたものとして 2009 年に Vanish [9] が提案された. Vanish ではデータそのものを消すわけではなくデータを復号する鍵を消去することによって一定期間たったデータが誰からもアクセスできなくなることを実現している. 具体的には DHT に復号鍵を秘密としたシェアを格納し, データの復号の際にシェアを取り出すことで復号鍵を復元するという方法をとっている. これにより, 復号鍵を管理する機関に信頼を必要としなくなった. しかし, Vanish の問題点としてシビル攻撃に弱いということが知られている. 実際に [10] では hopping 攻撃によって効率的な攻撃が示された. その後, hopping 攻撃に耐性のある Vanish として SafeVanish [11] が提案されたが, 依然としてシビル攻撃そのものにはぜい弱であることが指摘されている [12].

Vanish が抱えている根本的な問題として, DHT 側の要求と Vanish の要求が正反対であるということがある. DHT では信頼性を向上させるためにデータの複製が行われるが,

© 2017 Information Processing Society of Japan

それが Vanish への攻撃のし易さにつながってしまう．一方でファイル共有アプリケーションにはファイルの機密性ではなくファイルの格納や取り出しにオーバーヘッドができるだけ少ないことが求められる．したがってデータの複製をしない DHT は普及しにくく，それに伴って DHT に参加してくれるノードも少なくなってしまい，結局のところ Vanish のセキュリティの向上は望めない．

Vanish の根本的な問題であるシビル攻撃へのぜい弱性に解消したと主張する研究としては EphPub [19] がある．EphPub は DNS キャッシュリゾルバが鍵管理機関の役割を果たす．送信者は大量の DNS キャッシュリゾルバにドメインとドメインの有効期間を登録して一時的にビットを埋め込む．受信者はキャッシュリゾルバからビットを取得し，鍵を復元する．EphPub の問題点は送信者が最初に大量の DNS キャッシュリゾルバとドメインとドメインの有効期間のリストを持っている必要があるということにある．つまり，政府やハッカーの保有するキャッシュリゾルバがリストに紛れこむ可能性があるので，セットアップに強い信頼を必要とする．したがって EphPub における鍵管理機関である DNS キャッシュリゾルバは非中央集権ではなく分散的であり，ビットを保存する場所は EphPub の開発者が決めることができてしまう．

最後に，Vanish と EphPub には送信者が復号鍵を鍵管理機関に送るときに，送信者と鍵管理機関間の通信が守られておらず，シェアの盗聴や改ざんにつながる問題がある．EphPub でスタブリゾルバとキャッシュリゾルバ間の通信で DNS over TLS [20] を用いることも考えられるが，まだ実験段階の技術であり，普及しているとはいえない．

5. 提案手法

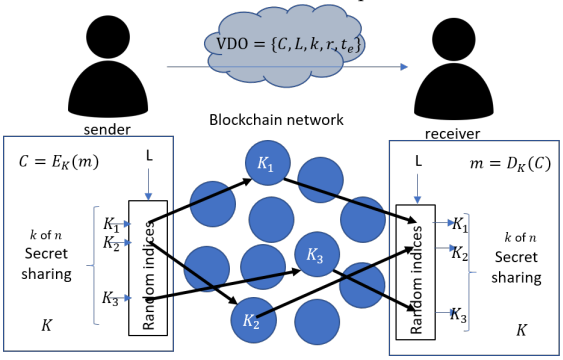
3 節では retroactive privacy を達成するために鍵管理機関が必要であることを述べたが，本研究で提案するプロトコルでは鍵管理機関としてブロックチェーンを利用する．これによって，鍵管理機関に対する信用を必要とせずシビル攻撃に対して耐性を持つこととなる．技術的にはブロックチェーンと秘密分散法を組み合わせたものとなる．

5.1 プロトコル手順概略

提案するプロトコルの概要図は図 1 で表される．送信者は期限付きデータをまず暗号化する．暗号化に使った鍵はシェアに分割され，ブロックチェーンネットワーク上のノードに安全な通信によって送られる．鍵を預けるのに単一のサーバーを使わず，ブロックチェーンのノードを用いるのでユーザーが必要とする信頼は分散され，単一障害点ともならない．シェアを送るノードはマイニングを行いかつ計算能力があればあるほど選ばれやすくなる．ブロックにはマイニングを行ったノードの公開鍵が入っており，チェーンは公開鍵暗号基盤としての役目を果たす．ブロックチェーンに埋め込まれたデータは non-equivocation の性質を満たすことが知られており [21]，ブロックチェーンは公開鍵暗号基盤として望ましい性質を有する．

そして，データの期限時刻がきたらブロックチェーンネットワーク上のノードはすみやかに自らの持つシェアを削除する．これにより，$t_e + \Delta$ 以降のどの参加者によるアクセスも不可能になる．

図 1　Overview of the protocol



5.2 プロトコル詳細

この節では提案の期限付きデータを消去するプロトコルについてより詳細に説明する．プロトコルでは参加者として送信者 S，受信者 R，ブロックチェーンネットワーク BN が存在する．S は VDO を送るクライアント，R は S から VDO を受け取るクライアントである．BN_i はブロックチェーンの i 番目のノードであることを表す．プロトコルは以下のようになる．

5.2.1 送信者プロトコル

S はランダムに共通鍵 K，乱数 r を選ぶ．そして，作りたいシェアの数と閾値をそれぞれ n と k を $n \geq k$ を満たすように決定し，データの期限時刻 t_e を定める．

次にメッセージ m をカプセル化する手順について説明する．まず，メッセージ m を共通鍵 K で暗号化して暗号文 $C = \mathsf{Enc}_K(m)$ を計算する．そして，現在のブロックチェーンネットワークの正しいチェーンである $chain$ から先頭 d 個のブロックを抽出する．得られたブロックの中から作りたいシェアの数 n だけランダムにブロックを選択し，選んだブロックに書き込まれている公開鍵 pk と IP のペアの合計 n 個を L とする．以上で得られた C, L, k, r, t_e をまとめて $VDO = \{C, L, k, r, t_e\}$ とする．

さらに，共通鍵 K から Shamir の秘密分散法を用いてシェアの列 $\mathbf{K} = K_1, K_2, \ldots, K_n$ を計算する．そして，暗号学的ハッシュ関数 $H(\cdot)$ を用いて $IP_i + r$ のハッシュ値 $H(IP_i + r)$ を計算し，キーとする．キーを $H(IP_i + r)$ として K_i, t_e をバリューとして BN_i に対して TLS プロトコルを用いて安全な通信路を介して送る．その際，BN_i の持つ公開鍵 pk_i を用いる．

© 2017 Information Processing Society of Japan

最後にセキュアなチャネルを介して R に VDO を送ることで S のプロトコルは完了する．以上の送信者プロトコルはアルゴリズム 1 のような疑似コードとして記述される．

Protocol 1 Sender - encapsulate message and send key shares

Require: message $m \in \{0,1\}^*$ and expiration time t_e
Ensure: send VDO and shares
1: pick $K \leftarrow_\$ \{0,1\}^k$ ▷ Initialization Phase
2: pick $r \leftarrow_\$ \{0,1\}^k$
3: pick n, k where $n \geq k$
4: $chain \leftarrow Retrievechain()$
5: $C \leftarrow \mathsf{Enc}_K(m)$ ▷ Message encapsulation Phase
6: $L \leftarrow ((pk_1, IP_1), \ldots, (pk_n, IP_n)) \leftarrow_n head_d(chain)$
7: $VDO \leftarrow \{C, L, k, r, t_e\}$
8: $\mathbf{K} = MultiShare_{sss}(n, k, K)$ ▷ Send shares Phase
9: **for** $i \leftarrow 1, n$ **do**
10: send $(H(IP_i + r), (\mathbf{K}[i], t_e))$ to IP_i using authentic channel
11: **end for**
12: send VDO to receiver using secure channel

5.2.2 受信者プロトコル

R は送信者から受け取った VDO の L と r を使用する．L の 1 番目から n 番目の要素について順番にブロックチェーンのノード BN_i に IP_i と BN_i の持つ pk_i を用いてセキュアな通信を行い，その後キー $H(IP_i + r)$ を用いてバリューの要素である鍵のシェア K_i を受け取る．

受けとった $K_i (i = 1 \ldots n)$ の列から k 個を選び Shamir の秘密分散法を用いて元の鍵 K を復元する．さらに，R は共通鍵 K を使って VDO の要素である暗号文 C から元のメッセージ $m = \mathsf{Dec}_K(C)$ を復元する．以上の受信者プロトコルはアルゴリズム 2 に疑似コードとして表現される．

Protocol 2 Receiver - load key shares and decapsulate message

Require: VDO
Ensure: message m
1: Initialize vectors to be **K'**, **K"**
2: **for** $i \leftarrow 1, L.length$ **do** ▷ Load shares
3: send $(H(IP_i + r))$ to IP_i using authentic channel
4: receive K_i from IP_i using authentic channel
5: **K'**$_i \leftarrow K_i$
6: **end for**
7: **K"** $\leftarrow_k$ **K'** ▷ recover K and Decapsulate VDO
8: $K' \leftarrow MultiReconstruct_{sss}$(**K"**)
9: $m \leftarrow \mathsf{Dec}_{K'}(C)$

5.2.3 鍵管理プロトコル

ブロックチェーンのノードは Bitcoin のように通常のマイニング作業を行うが，ブロックの要素が Bitcoin とは少し異なる．それに加えてマイニングとは別に https 通信を行うサーバーとしての機能を持つ．

提案するプロトコルにおけるブロックチェーンでは，ブロックを B とすると，$B = \{s, ctr, pk, IP, \mathbf{tx}\}$ である．ブロックの要素にブロックの生成者の公開鍵と IP アドレスが加わったのが Bitcoin と異なる点である．

次に鍵管理機関としてのプロトコルであるが，最初にノード起動時に自身のハッシュテーブル T を初期化する．外部からキー key とバリュー $value$ のペアが送られたら該当のキーを探しバリューを上書きする．外部からキー key のみが送られたらキーに対応するバリューを探し，存在したらバリューを外部に渡す．最後にハッシュテーブル内に存在するキーについて期限時刻がきたら該当のキーを削除する．以上のブロックチェーンノードによる鍵管理プロトコルはアルゴリズム 3 に疑似コードとして表される．

Protocol 3 Blockchain node BN_i - manage key shares

Require: public key $\mathsf{pk}_i \in \{0,1\}^n$, IP address IP_i
1: **On Initialization:**
2: Initialize hash table to be T.
3: **On receive** $(key, value(= (share, expiration\_time)))$ **via authentic channel:**
4: $T_{key} \leftarrow value$
5: **On receive** key **via authentic channel:**
6: return $T_{key}.share$
7: **Ongoing:**
8: **for** $key \leftarrow 1, T.length$ **do**
9: **if** $t \geq T_{key}.expiratioin\_time$ **then**
10: **delete** T_{key}
11: **end if**
12: **end for**

6. 安全性の分析

6.1 暗号の危殆化

鍵による消去の手法によってデータを消去することの問題点として，多項式時間の計算能力を有する攻撃者を仮定しないと暗号文単独攻撃によって暗号文が復号されてしまうということがある [22]．この問題に対して [23] は暗号が危殆化した場合でも暗号化されたデータが復号されないような仕組みを提案した．具体的な方法としては共通鍵だけでなく暗号文の一部をシェアとすることによって，暗号が危殆化しても暗号文の断片からは平文が復元されないようにするといったものである．この手法を本研究で提案しているプロトコルに組み込めば暗号の危殆化に対して耐性を持つことができる．

6.2 ブルートフォース攻撃と DoS 攻撃

シェアを得るために第三者がノードに対して手当たり次第にキーをリクエストするブルートフォース攻撃の可能性が考えられる．また，それと同じ方法でノードに対して処理能力を超えるリクエストを投げてサービスを停止させる DoS 攻撃の可能性も考えられる．前者はプロトコルの retroactiev privacy，後者は可用性に対する攻撃となる．

© 2017 Information Processing Society of Japan

まず前者のブルートフォース攻撃についてはシェアを得る確率はハッシュテーブルのキーの空間の大きさに反比例する. 例えばキーの空間の大きさを 2^{128} としてしまえば, AES の 128bit 長の鍵を総当たりで探すのと難易度は定数倍しか変わらない.

次に DoS 攻撃であるが, 既に PieceWork [24] という研究が存在する. PieceWork は計算の一部を小さいパズルとして外部にアウトソースできる Proof-of-Work (PoW) である. つまり, 外部からリクエストが来た場合に小さいパズルを解かせるようにすれば, 攻撃者は計算資源を消耗することになり外部からの DoS 攻撃を緩和できる.

7. 実装と評価

ブロックチェーンのノードと送信者と受信者のクライアントそれぞれについて実装を行った. 使用言語は全て C++ である.

7.1 ブロックチェーンのノード

Bitcoin Core クライアント [25] のバージョン 0.14 に行った. Bitcoin Core クライアントは P2P 通信用のサービスと RPC 通信用のサービスを有しているが, 今回は送信者や受信者と通信を行い, シェアのやり取りを行う機能を追加するために https 通信を行うサービスを新たに実装した. Bitcoin Core クライアントへの実装では既にクライアントで使われていた以上のライブラリを新たに必要とすることはなかった. ただし, OpenSSL [26] のバージョン 1.1 以降で使用可能な関数である *TLS_server_method* を用いている.

7.1.1 ブロックヘッダーの要素

Bitcoin におけるブロックの要素に加えて新しく秘密鍵のハッシュ値（32 バイト）, IP アドレス（4 バイト）, ポート番号（4 バイト）の要素を追加した. 秘密鍵のハッシュ値はノードが https 通信で使用する秘密鍵のハッシュ値をとったものである. ハッシュ関数のアルゴリズムは SHA-256 を使用している. 次に IP アドレスはブロックをマイニングしたノードのグローバル IP アドレスである. 最後のポート番号についてはデバッグするための要素であるが, ノードが https サーバーとして動くときに使用するポート番号を表す.

7.1.2 https サーバーの API 構成

https サーバーにおけるリクエストの形式は 2 種類ある. 一つはシェアを格納するリクエスト, もう一つはシェアを受け取るリクエストである. シェアを格納するリクエストの形式は以下で表すことができる.

- リクエストの種類 - 2 バイト
- キー - 4 バイト
- シェアの長さ - 4 バイト
- シェア - シェアの長さの領域の値による
- 期限時刻 - 4 バイト

リクエストの種類は今回の実装では 2 種類であるが, シェアを格納するリクエストに関しては値を 1 とした. キーはハッシュテーブルのキーを表す. そして, シェアの長さについては値がシェアのバイト数を表し, シェアの領域はその分の長さだけリクエスト内のフィールドが確保される. 期限時刻は UNIX 時間で表される.

次にシェアをノードから受け取るリクエストについてであるが, リクエストの形式は以下のようになる.

- リクエストの種類 - 2 バイト
- キー - 4 バイト

シェアをノードから受け取るリクエストについては**リクエストの種類**の値を 2 とした. シェアをノードから受け取るリクエストに対するレスポンスは以下のように設計した.

- シェアの長さ - 4 バイト
- シェア - シェアの長さ領域の値による

7.2 送信者と受信者のクライアント

送信者と受信者のクライアントについては共通鍵による暗号化や復号のために Crypto++ライブラリ [27] 5.6.5, https 通信のために OpenSSL バージョン 1.1 を用いている. また, Shamir の秘密分散法は自前で実装を行った.

7.3 評価

今回の評価では送信者プロトコルが全て完了するのにかかる時間, 受信者プロトコルが全て完了するのにかかる時間のそれぞれついて計測を行った.

評価環境として Amazon Web Services (AWS) が提供している EC2 インスタンスをシードとし, 加えて東京大学生産技術研究所のノードと東邦大学のノードの合計三つのノードで実験を行った. 全てのノードが regtest モードで起動してあり, Bitcoin Core クライアントの提供しているブロックの生成コマンドによって生産技術研究所によって作られたブロックが 100 個, 東邦大学によって作られたブロックは 100 個である. シードである AWS のノードはシェアを保存する役割は持たない. そして, シェアの数 $n = 2$, 閾値を $k = 2$ として送信者プロトコルの完了と受信者プロトコルの完了にかかる時間を計測した（表 1）.

表 1 Amount of time required for an execution of sender protocol or receiver protocol

	Sender protocol	**Receiver protocol**
time[s]	0.004873	0.002214

計測した結果は送信者プロトコルの完了に 0.004873 秒かかり, 受信者プロトコルの完了に 0.002214 秒かかった. 実際のシステムの使用では, シェアの数を増やしたりノードの接続状況が悪いと, 送信者プロトコルや受信者プロトコルにかかる時間は増えると考えられる.

© 2017 Information Processing Society of Japan

また, 今回の評価環境ではブロックの数は 200 個であったが, Bitcoin ではブロックの数はおよそ 48000 あり, 実際にプロトコルを動かす際はブロックヘッダーの同期の部分が最も時間のかかる箇所であると考えられる. そこで Bitcoin におけるブロックヘッダーの同期時間の実測を行った. ライブラリは libbtc [28] を用いた. チェーンのチェックポイントはブロックの高さ 403200 で, 最新ブロックの高さである 482310 個までを 103.40Mbps の回線速度で同期する. 全くブロックヘッダーを同期していない状態から同期を開始した場合, 同期完了にかかる時間は 12.400 秒であった. 一方で, ブロックヘッダーの同期が完了している状態から再びブロックチェーンネットワークと接続し, 同期完了したことがわかるのにかかる時間は 0.543 秒であった.

この結果から送信者のクライアントの最初のセットアップには時間がかかるが, 一度同期が完了してしまえば再びプロトコルを実行したときにかかる時間は短いといえる.

8. 結論

本研究ではシビル攻撃に耐性のある, 鍵管理機関に対する信用を必要としないデータの自動消去プロトコルを提案した. 今回は期限時刻にシェアが消えることによってメッセージが復号できなくなると説明したが, このプロトコルでは送信者がシェアの格納されている場所を把握できるので, 期限時刻より前に新しいリクエストを発行してノードに保存されているシェアを別の値で上書きすることで期限時刻より前にメッセージを復号できなくする柔軟性も併せ持つ.

今後の課題として, シェアを受け取ったノードが第三者にシェアを渡さないような仕組みを考える必要がある. しかし, データを第三者に渡さないことを強制するのは非常に難しい. したがってシェアを第三者に渡す場合に何らかのペナルティーをノードに対して与える仕組みを考えることが最も自然であるだろう.

謝辞 実験に協力頂いた東邦大学理学部情報科学科准教授 金岡先生に感謝する． 本研究の一部は JSPS 科研費 17KT0081 の助成を受けた.

参考文献

[1] “日本国憲法.” law.e-gov.go.jp/htmldata/S21/S21KE000.html.
[2] 宍戸常寿, “通信の秘密について,” 2013.
[3] “Australia’s plan to force tech giants to give up encrypted messages may not add up.” https://www.theguardian.com/technology/2017/jul/14/forcing-facebook-google-to-give-police-access-to-encrypted-messages-doesnt-add-up.
[4] R. Singel, “Encrypted e-mail company hushmail spills to feds.” https://www.wired.com/2007/11/encrypted-e-mai/, 2007.
[5] “How the NSA Almost Killed the Internet.” https://www.wired.com/2014/01/how-the-us-almost-killed-the-internet/.
[6] J. CHENG, “Over 3 years later, “deleted” facebook photos are still online — ars.” https://arstechnica.com/business/2012/02/nearly-3-years-later-deleted-facebook-photos-are-still-online/, 2012.
[7] M. Rex, “Deleted cloud files are not always gone permanently – celebrities learn the hard way.” http://insights.wired.com/profiles/blogs/deleted-cloud-files-are-not-always-gone-permanently-celebrities, 2014.
[8] R. Gallagher, “British authorities demand encryption keys in case with “huge implications”.” https://theintercept.com/2016/04/01/british-authorities-demand-encryption-keys-in-closely-watched-case/, 2016.
[9] R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy, “Vanish: Increasing data privacy with self-destructing data,” in *18th USENIX Security Symposium, Montreal, Canada, August 10-14, 2009, Proceedings* (F. Monrose, ed.), pp. 299–316, USENIX Association, 2009.
[10] S. Wolchok, O. S. Hofmann, N. Heninger, E. W. Felten, J. A. Halderman, C. J. Rossbach, B. Waters, and E. Witchel, “Defeating vanish with low-cost sybil attacks against large dhts,” in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2010, San Diego, California, USA, 28th February - 3rd March 2010*, The Internet Society, 2010.
[11] L. Zeng, Z. Shi, S. Xu, and D. Feng, “Safevanish: An improved data self-destruction for protecting data privacy,” in *Cloud Computing, Second International Conference, CloudCom 2010, November 30 - December 3, 2010, Indianapolis, Indiana, USA, Proceedings*, pp. 521–528, IEEE Computer Society, 2010.
[12] L. Zeng, S. Chen, Q. Wei, and D. Feng, “Sedas: A self-destructing data system based on active storage framework,” in *APMRC, 2012 Digest*, pp. 1–8, IEEE, 2012.
[13] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” http://bitcoin.org/bitcoin.pdf.”
[14] F. Hao, D. Clarke, and A. F. Zorzo, “Deleting secret data with public verifiability,” *IEEE Trans. Dependable Sec. Comput.*, vol. 13, no. 6, pp. 617–629, 2016.
[15] D. Boneh and R. J. Lipton, “A revocable backup system,” in *Proceedings of the 6th USENIX Security Symposium, San Jose, CA, USA, July 22-25, 1996*, USENIX Association, 1996.
[16] R. Perlman, “The ephemerizer: Making data disappear,” tech. rep., Mountain View, CA, USA, 2005.
[17] R. Dingledine, N. Mathewson, and P. F. Syverson, “Tor: The second-generation onion router,” in *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA* (M. Blaze, ed.), pp. 303–320, USENIX, 2004.
[18] S. Khandelwal, “Tor network is under attack through directory authority servers seizures.” http://thehackernews.com/2014/12/tor-network-hacked.html, 2014.
[19] C. Castelluccia, E. D. Cristofaro, A. Francillon, and M. A. Kâafar, “Ephpub: Toward robust ephemeral publishing,” in *Proceedings of the 19th annual IEEE International Conference on Network Protocols, ICNP 2011, Vancouver, BC, Canada, October 17-20, 2011*, pp. 165–175, IEEE Computer Society, 2011.
[20] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, and D. Wessels, “P. hoffman,” specification for dns over transport layer security (tls),” tech. rep., RFC 7858, DOI 10.17487/RFC7858, May 2016,¡ http://www.rfc-editor.org/info/rfc7858.
[21] A. Tomescu and S. Devadas, “Catena: Efficient non-equivocation via bitcoin,” in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 393–409, IEEE Computer Society, 2017.
[22] J. Reardon, D. A. Basin, and S. Capkun, “Sok: Secure data deletion,” in *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pp. 301–315, IEEE Computer Society, 2013.
[23] G. Wang, F. Yue, and Q. Liu, “A secure self-destructing scheme for electronic data,” *J. Comput. Syst. Sci.*, vol. 79, no. 2, pp. 279–290, 2013.
[24] P. Daian, I. Eyal, A. Juels, and E. Gün Sirer, “(short paper): Piecework: Generalized outsourcing control for proofs of work,” 2017.
[25] B. C. Developers, “Bitcoin core.” https://bitcoin.org.
[26] E. A. Young, T. J. Hudson, and R. Engelschall, “Openssl: The open source toolkit for ssl/tls,” 2011.
[27] W. Dai, “Crypto++ library 5.1-a free c++ class library of cryptographic schemes.” http://www.cryptopp.com/, 2011.
[28] “libbtc.” https://libbtc.github.io/.

© 2017 Information Processing Society of Japan