マルチベンダ開発文書流通における P2P 型セキュリティ管理手法の設計

川島 悠太[1] 寺島 美昭[1]

概要：マルチベンダ開発における電子文書の共有管理は、データセンタに預けて各社がアクセスすることで閲覧を行うことが主流であるが、その管理手法では、データセンタに依存した管理となってしまう。データセンター依存の管理手法では有事の際に発行元が文書の安全管理に関わることが出来ないという問題点がある。本稿では、発行元が常に発行文書を管理できるような P2P 型の電子文書流通管理システムの設計について述べる。電子文書が複数回共有される「流通状況」において、P2P 型管理では情報漏洩などが危惧される。それらのリスクへのアプローチのため流通管理の要件を分析した。その要件を満たす管理手法を実現するためにブロックチェーンを用いる。これにより漏洩などのリスクを減らした実現性の高い P2P 型の流通管理システムの管理手法と設計を提案する。一部は実際に試作を行い、今後、流通管理システムを実現するための課題についての考察も述べる。

Design of P2P Security Management Method for Multi-Vendor Development Document Distribution

KAWASHIMA YUTA[1] TERASHIMA YOSHIAKI[1]

1. はじめに

マルチベンダ開発における文書管理では、データセンタに文書を送信し、データセンタに向けて各企業がアクセスすることが一般的である。しかし、データセンタにセキュリティ的な問題がある場合や、攻撃をうけた場合には、発行元企業は文書を守ることができない。企業の開発文書等は機密要求が高く、文書が漏洩した場合は発行元企業だけでなく、開発に関わった企業にも損害が出るケースがある。そのためデータセンタに頼った管理手法では、発行元の努力が及ばない部分で文書が脅かされる可能性がある。逆にデータセンタを用いない方法を、現在の日本でも頻繁に行われている下請け開発に当てはめると、文書の共有状況の追跡が行えず、共有先に悪意あるユーザが混ざっていた場合には情報漏洩に繋がる (図 1)。

本稿では下請け開発のような、文書が二次、三次と共有されていくマルチベンダ開発において第三者機関を介さず、開発に関わる企業のみで P2P 型の管理を行うためのシステム設計について述べる。各企業は自身が発行した文書に対して、発行元企業が管理主体となり、複数の企業が自身の発行した文書については相互的に管理を行う (図 2)。発行元企業は文書の共有状況や、閲覧日時等をモニタリングし、なんらかのトラブルが会った際には文書の閲覧をていしできるような管理設計を行う。

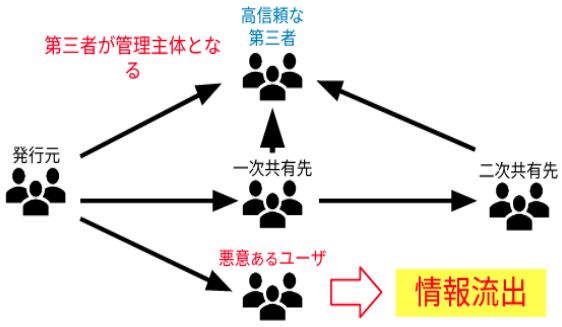

図 1 P2P 型管理の問題点

[1] 創価大学大学院工学研究科情報システム工学専攻

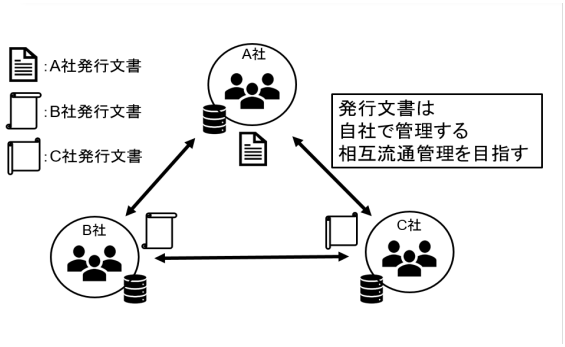

図 2　文書発行企業による文書の相互管理

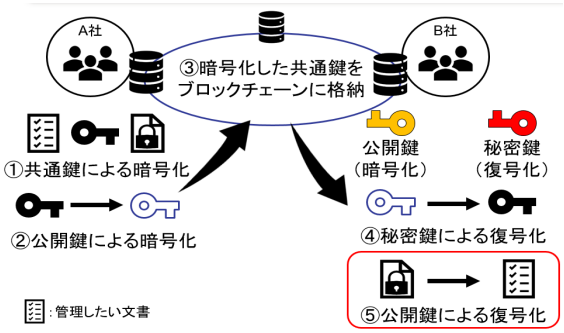

図 3　ブロックチェーンによる鍵配送 (先行研究)

P2P 型の流通管理設計を実現するために、Bitcoin などで用いられるブロックチェーンを用いる。P2P 型の文書管理は現状では多く普及していないが、P2P 型の通貨管理としてはブロックチェーンは非常に一般的な管理技術として用いられる。通貨というシビアなデータを管理できるのであれば、それを文書に適用することができれば、通貨と同じレベルのセキュリティを担保できる。Bitcoin においてはブロックチェーンは通貨の送金履歴や、アカウントの残金の把握のためのデータ管理に用いられる。ネットワーク内に共有される履歴情報はトランザクションと呼ばれる。ブロックチェーンには、管理する情報に対して、トレーサビリティの高い管理、デジタルデータの共有による管理情報の透明性、改ざんが困難であるという情報の信頼性、分散システムのためシステムダウンが起こりにくいというメリットがある。また、ネットワーク参加者にはアドレスが割り振られるため個人 (企業) の特定が容易に行える。後述する流通管理要件に対して、これらの特性が非常に有用であると判断し、本稿ではブロックチェーンを用いた P2P 型の流通管理システムにおける管理手法について述べる。管理のためセキュリティ要件を分析し、それを満たす設計を行う。

2.　関連研究

本研究と同様のアプローチを行っている企業間の電子文書管理の研究として、日本電信電話株式会社の近田らは社内インフラで用いられる暗号化ソリューションとブロックチェーンを組み合わせた管理手法を提案している。先行研究では、暗号化した文書の復号鍵の共有をブロックチェーンで行うことで文書の管理を実現している。これによって第三者機関を介さずに証拠製の高い公平な記録が残る 1 対 1 の文書共有を実現している。しかし、近田らの研究を本研究の文書流通状況に持ち込んだ際、同一の暗号化ソリューションを用いる必要があるという導入コストの高さが問題となる。また、文書は発行元からの直接共有しか行えず、共有先から更に共有先に文書が共有されるなどの、日本の開発において頻繁に現れる下請け開発への適用が難しい。

3.　システムモデル

流通状況とは文書が発行元から、共有先、そして共有先から別の企業に共有されるような状態を指す (図 4)。この状況において文書の発行元は自分の発行文書がどのように共有されて、いつ見られたのかなどの履歴情報を常にモニタリングすることで発行文書の管理を行う。共有の際には共有履歴を残すことで、ネットワーク参加者にはなんらかの文書が共有されたことが伝わるため、データの送信ミスによるトラブルを防止することができる。また文書データをブロックチェーン上で管理することで、発行元のサーバやネットワークが停止した場合でも文書の閲覧を行うことができる。

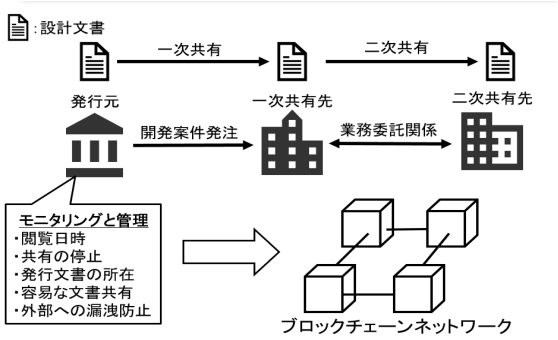

図 4　ブロックチェーンを用いた本研究のアプローチ

4.　提案手法

電子文書の流通管理を行うために、流通状況における文書のセキュリティ要件を分析した。流通管理をするために管理しなければならない情報は、発行元アドレス、共有先アドレス、閲覧や共有における日時情報、文書データである。これらを管理するためには以下の 4 点を満たすことが必要である。提案では、これらを解決する流通文書の管理手法について述べる。

- **記録性**　閲覧記録や流通情報の履歴が残ること
- **一貫性**　流通している文書が全て同一であること
- **機密性**　許可されたユーザのみしか閲覧ができなこと
- **流通性**　二次、三次の共有が行えること

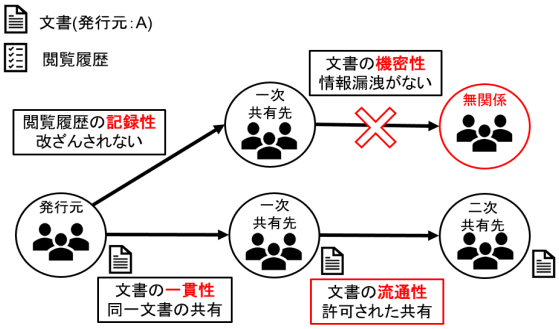

図 5　流通管理におけるセキュリティ要件

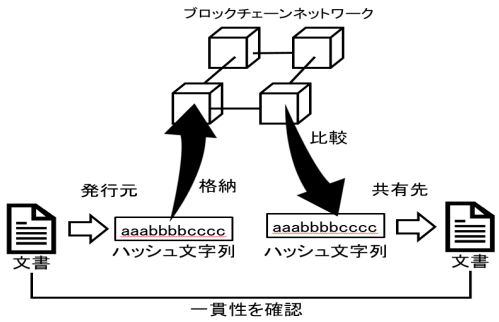

図 7　共有時のハッシュ比較

4.1　記録性の解決

記録性は履歴情報の発行により解決する。ブロックチェーンを用いることで、ブロックチェーンに格納された履歴データに高い改ざん耐性をもたせることができる。共有時や閲覧時にブロックチェーンに取引履歴を格納することで記録性を満たすことができる。管理する情報は、共有時は発行元アドレスと共有先アドレス、文書データを格納する。閲覧時は閲覧日時と閲覧者アドレスを格納することで共有と閲覧における発行元のモニタリングを実現する。

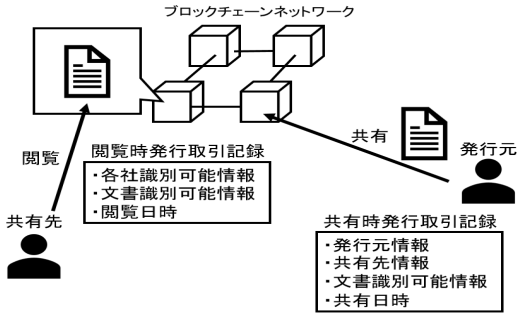

図 6　ブロックチェーンによる履歴管理

4.2　一貫性の解決

一貫性はハッシュ暗号によって解決する。流通されている文書はバージョンなどの齟齬なく同一の文書が共有されていなければならない。もし文書が異なっている場合、共有されている情報に齟齬が生じるためマルチベンダ開発等の規模の大きい開発では大幅な手戻り等の原因になる。ブロックチェーンは元々データの肥大化が懸念されており、ブロックチェーンが構想された初期からハッシュは利用されているほど相性が良いため、文書を共有する際に、共有文書のハッシュを計算する。これをブロックチェーン上に最新版として格納し、共有先はそのハッシュ文字列と自身が所持している共有文書のハッシュ文字列を比較することで共有文書が同一であるかを確認することができる。これにより共有されている文書の一貫性を満たす。

4.3　機密性の解決

文書の難読化と認証機能で機密性を解決する。ブロックチェーンでは同一の情報が全てのネットワーク参加者に共有されてしまうため、共有している文書の閲覧を制限する必要がある。共有の際に共有先のアドレスを入力し、それをブロックチェーン上に保持することで、閲覧要求が発行された場合に共有されているネットワーク参加者のアドレスと今回閲覧要求を出したネットワーク参加者を照会することで文書の閲覧が許可されているのかを判定することができる。

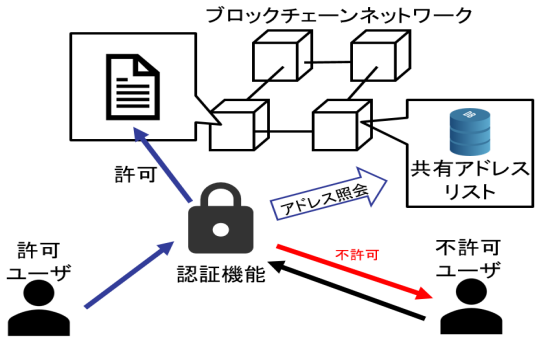

図 8　アドレスを利用した認証機能

しかし、これだけでは文書がネットワーク参加者全員のサーバへ共有されてしまい、認証機能を介さずに文書の閲覧を行うことが可能になってしまうため、文書に難読化の処理を施す。文書の共有の際に、変換、分割、格納の処理を行う。文書データは概ねバイナリデータで管理されているため、それを Base64 などの文字列に変換し、出力された文字列を一定サイズに分割する。その分割された文字列にランダムな ID を割り振る。このランダムに振られた ID を文書の形式に復号できるように繋げ合わせるための復号リストを作成し、共有先と発行元だけが保持する。これにより機密性を満たす。

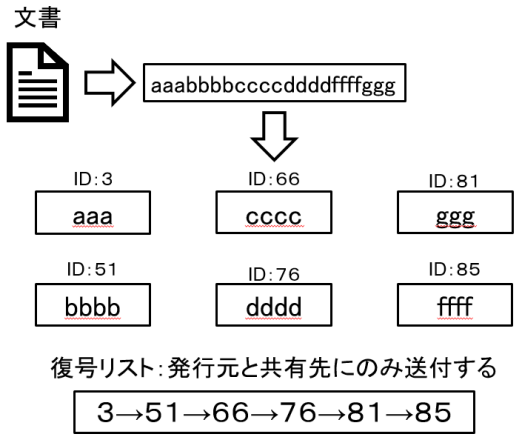

図 9　文書の難読化処理

4.4　流通性の解決

文書を通貨形式 (以下、トークン) に変換して管理する (図 10)。文書が 2 次、3 次と共有されるためには、発行元を介さずに共有可能にする必要がある。4.3 節で述べたように文書には難読化処理を施すため、その分割した文書文字列をトークンに変換し流通させ管理する。トークンには ID を振ることができるものを利用し、4.3 節で述べた難読化処理でランダムに振り分けた ID をトークンに割り振る。閲覧の際には所持しているトークンから文字列を取得し復号リストに従ってつなぎ合わせることで閲覧が可能になる。

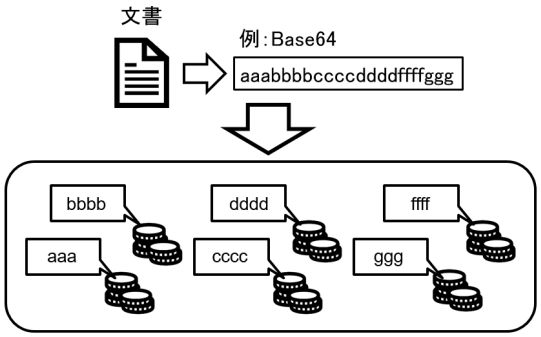

図 10　文書を分割トークンに変換することで流通性を解決

以上の手法によって、管理要件として挙げた 4 項目を満たしつつ、電子文書の流通管理を実現することができる。

5.　システム設計

4 章の管理手法を踏まえたシステム設計について述べる。分割した電子文書文字列の保持と ID の割り振りが行えるトークンの規格として Ethereum で利用することができる ERC721 規格を用いる。この規格はトークンとしてユーザ間でやり取りができる通常の通貨機能だけでなく、トークン 1 枚ごとに ID を設定し各トークンが一定量のデータを保持することが可能である・また同一のプログラムからトークンは生成されるが、それぞれのトークンが別々の価値を持つことができるため代替不可能トークンと呼ばれる。これを用いるためにブロックチェーンネットワークは Ethereum を使用する。ERC721 規格のトークン設計を行う場合には、Ethereum 開発言語である Solidity の openzeppelin というライブラリを用いて開発することが推奨されているためそれを用いてトークンの設計を行う。Ethereum を用いることでブロックチェーン上にプログラム (以下、コントラクト) を配置し動作させることができる。Ethereum でコントラクトを動作させるためには「gas」と呼ばれる手数料を支払う必要がある。この gas の最大量を定義する gaslimit という数値が Ethereum におけるブロックサイズを表している。そのため、gaslimit の数値を大きく設定するほど、複雑な処理をブロックチェーン上で行うことができるが、ブロックサイズが大きくなっていく。

マルチベンダ間を流通する電子文書を管理する場合、不特定多数のネットワーク参加者がいるメインネットで文書データを管理する必要性は少ない。そのためマルチベンダ開発に携わる企業のみでプライベートネットワークを構築し、その中で文書の流通を管理する。プライベートネットワークを利用することでブロックサイズに関係する gaslimit などの数値を、扱うデータに適した数値に変更することができるようになり、システム設計の際の自由度も飛躍的に増加する。本稿の設計では 1 億に設定した。仮に、プライベートネットワークでは信頼性が担保しきれていない場合は、本システムをサイドチェーンとして一定期間ごとにブロックチェーン自体をハッシュ化して、そのハッシュをメインネットに格納していくことで、履歴データの改ざんの防止をより強固にすることができる。

5.1　使用 OSS

表 1　使用する OSS と機能

OSS	機能
ganache-cli	プライベートネットワークの構築
truffle	コンパイルと ganache-cli へのデプロイ
Metamask	ganache-cli の秘密鍵の管理と取引記録の発行

本研究では、実現性の高いシステムを構築するために Ethereum 開発において多く用いられる 3 種の OSS を用いた。まず、ブロックチェーンネットワークの構築には「ganache-cli」を用いてプライベートネットワークの構築を行う。2 つ目は ganach-cli と同一の企業が開発をした「truffle」を用いる。truffle では独自で開発したコントラクトのテストとコンパイル、ganache-cli へのデプロイを行う。3 つ目は秘密鍵の管理と取引記録の発行のため「Metamask」を用いる。Metamask は Firefox と Chrome の拡張機能であるため、本研究で使用するブラウザは Firefox か Chrome に絞る。

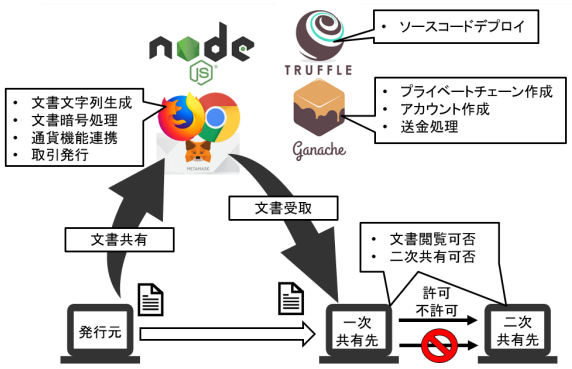

図 11　OSS を用いたシステムの詳細モデル

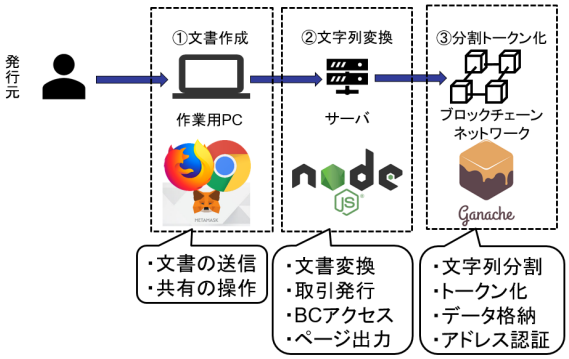

図 12　共有フローと使用インフラ

5.2　設備と共有フロー

本システムを利用するために各企業が用意する設備と、文書の共有フローについて述べる。企業ごとに必要な設備は、プライベートネットワークに接続し、企業のアドレスを保持するサーバである。試作では Ethereum 開発に用いられる Web3 というライブラリが javascript 対応なので同一の言語で記述することができる Node.js を利用してサーバを設計した。

具体的な共有の処理について述べる。

(1) 発行元が共有する文書を発行
(2) ブラウザを介してサーバへ送信し文字列へ変換
(3) 共有取引を発行
(4) プライベートネットワークへ文字列を送信
(5) 文字列を分割し、トークン化
(6) 共有先はトークンから文書文字列を取得
(7) 復号リストに従い文書文字列を復号
(8) バイナリデータへ変換して閲覧

設備と OSS の対応は図 12 のようになる。各社がプライベートネットワークにアクセスできるサーバで ganache-cli を運用し、トランザクションの発行は Node.js で行う。Node.js サーバは HTML ページの運用や文書の文字列変換、トランザクションの発行の役割を持つ。ganache-cli 上で動作するコントラクトは文書文字列の格納、分割、トークン化などの役割を持つ。

6.　試作開発結果

文書をプライベートネットワークに送信し、それを取得するまでの処理を試作した。文書を文字列に変換し、Metamask でトランザクションを発行する。コントラクト上で文書文字列の分割処理を行い、それを取得する。共有時の挙動を実験的に確かめるため、共有処理に必要なデータを同時に格納する。今回の実験では、文書の文字列、発行元アドレス、共有先アドレス、文書文字列のハッシュ、ID を格納する。実験的に、18.3KB という PowerPoint1 枚を出力した PDF ファイル (以下、テストファイル) で実験を行う。

テストファイルの変換、分割の処理行いコントラクトが終了するまでにはおよそ 10 秒程度であった。テストファイルを Base64 形式に変換すると、24368 文字となったため、分割サイズは 10 分割として 2500 文字に設定した。全ての処理を完了して消費された gas は 35442954 となる。

データ量の小さいテストファイルではあるが、10 秒ほどで全ての処理を終えられることがわかった。今後はデータサイズや分割数などを変更して、明確なセキュリティ基準を設けて評価を行う。

7.　考察

7.1　ブロックサイズ

gaslimit 設定によるブロックサイズの適切なサイズ設定が課題となる。ブロックサイズの増加によるデメリットはブロックデータの伝達の速度とマイニングの集中にある。ただ、マルチベンダー間における流通状況を想定するのであれば、限られたユーザ数の環境で、マイニングによる報酬の発生もないため、それらの問題に大きな影響はないと判断して、今回の試作ではブロックサイズの増加設定を行った。

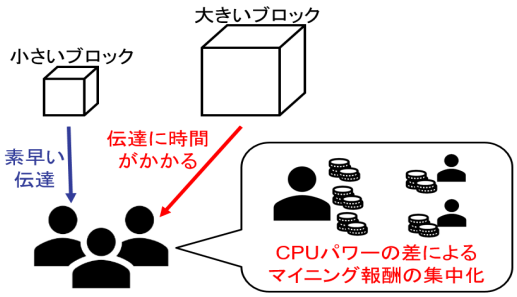

図 13　ブロックサイズ変更で起こる問題

今回は gaslimit を 1 億に設定して実験を行ったが。1 億という数値に確たる妥当性はない。テストファイルを格

納、ハッシュ化、分割までを実行しても余裕を持てる数値として暫定的に設定した。Ethereum のメインネットと比較するとかなり肥大したブロックサイズであり、ブロックデータの伝達速度に影響があるので、最小の gaslimit を設定したほうが良い。そのため今回のコントラクトで行った処理を分割してそれぞれの gas を計測した。単純な格納よりも、分割やハッシュ化などの複雑な処理のほうが消費される gas が高いことがわかった。それぞれの処理における gas の消費量の目安となるのでこれを元に gaslimit を小さく設定できるよう設計を向上していく。

表 2　処理ごとの gas 値

処理	消費した gas
Base64 形式の格納	768119
Base64 のハッシュ化	1081176
Base64 の 10 分割格納	20075199
共有情報の格納	15784873
分割と共有情報の格納	35442954

7.2　文書の分割保持

4.3 節で述べた機密性の解決において文書文字列の分割管理という方法を述べたが、トークン化して ID を割り振ったとしても、共有されている履歴情報から全探索で文書の復号可能となっている可能性があるため難読化だけでは機密性を高い水準で満たせていない。最も簡単な方法は鍵暗号通信による受け渡しだが、鍵の配送問題や鍵の管理方法など、別の問題も発生するため、文書の流通方法に関しては今後さらに検討を深めていく必要がある。

7.3　トランザクションの発行回数

トランザクションの関連付けが課題となる。 1 つの文書に対してトランザクションが複数回送信されると、トランザクション同士の関連を追いかける必要がある。常に同時刻に 1 社しか共有を行わない場合は容易に追跡を行うことができるが、複数の企業が同時に、様々な内容のトランザクションを発行する場合、追跡には別途アルゴリズムが必要となる。そのため、今回の設計ではコントラクトに分割の処理を行わせることでトランザクションの発行を 1 回で済むように設計した。 1 回の発行で処理を終えることができれば、共有されたデータの関連の追跡も容易に行える。しかし、トランザクションは gaslimit の影響を受けるため、複雑な処理を今後増やしていくのであれば、トランザクションを複数回に分けて実装することで 7.1 節のブロックサイズの問題を解決することが出来る。文字列の格納だけであれば、24368 文字で gas は 768119 で済むので、トランザクションの関連を管理するアプローチも検討する必要がある。

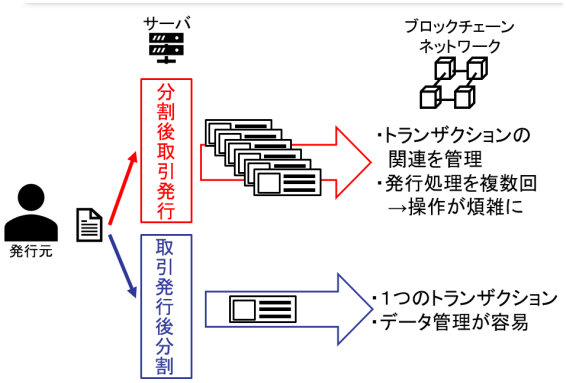

図 14　トランザクション発行のタイミング

8.　まとめ

本稿では、P2P 型の電子文書の流通管理システムの設計を提案した。P2P 型の文書流通管理におけるセキュリティ要件を分析し、それを満たすように設計を行った。また、実際に試作することを想定し、Ethereum 開発において頻繁に利用される OSS を用いることで実現性の高い設計を組み立てた。一部試作し、実験を行ったが、数値設定や手法にまだ検討の余地があることがわかったため、今後はより複雑なモデルや、発行元による管理機能の拡張など管理システムをより強化するようなモデルを設定し、試作を通して効果を確認していきたい。

参考文献

[1] 近田 昌義、他 6 名「ブロックチェーンを活用した組織間のドキュメント流通 」 日本電信電話株式会社 NTT サービスエボリューション研究所 電子情報通信学会信学技報　LOIS2019-03

[2] 大橋 盛徳、他 5 名「トークン連動型分散ファイルシステムの提案」 日本電信電話株式会社 NTT サービスエボリューション研究所 情報処理学会第 81 回全国大会

[3] 石田 達郎、他 4 名「ブロックチェーン上で柔軟なトークン設計によって実現するコンテンツ管理手法」 日本電信電話株式会社 NTT サービスエボリューション研究所 情報処理学会第 81 回全国大会

[4] 加嵜長門、篠原航「ブロックチェーンアプリケーション開発の教科書」マイナビ出版 2018 年

[5] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System, ” http://bitcoin.org/bitcoin.pdf ,2020/5/18 アクセス

[6] W. Entriken, D. Shirley, J. Evans, and N. Sachs, “ERC-721 Non -Fungible Token Standard, ” 2018 ,https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md . 2020/4/30 アクセス