卒業論文 **2018** 年度 **(**平成 **30** 年度**)**

ブロックチェーン技術とメッセージング技術を使用した **IoT** データ市場 **MIWWG** の提案と実装

指導教員

慶應義塾大学環境情報学部

中澤 仁
村井 純
楠本 博之
中村 修
Osamu Nakamura
Rodney D. Van Meter III
植原 啓介
三次 仁
高汐 一紀
武田 圭史

慶應義塾大学 総合政策学部
井上 義之

tigerman@ht.sfc.keio.ac.jp

学部論文要旨 2018 年度 (平成 30 年度)

ブロックチェーン技術とメッセージング技術を使用した IoT データ市場 MIWWG の提案と実装

論文要旨

日本語で要旨、ベタ書きで ok.

キーワード
ブロックチェーン, IoT, メッセージングシステム

慶應義塾大学総合政策学部
井上 義之

Abstract of Bachelor's Thesis Academic Year 2018

Implementing and Evaluating MIWWG : IoT data market which is made of blockchain and messaging technology.

Abstract

abstract in English.

Keywords
blockchain; IoT; messaging system

Keio University
Faculty of Policy Management
Yoshiyuki Inoue

目次

図目次

表目次

第 1 章

序論

本章では、最初に本研究における背景およびその現状の問題点を述べる。そのあと、これに対する本研究の目的とアプローチについて述べる。そして最後に、本論文の構成について示す。

1.1 背景

私たちの身の回りには様々な IoT 製品が存在している。その最たる例はスマートフォンであろう。Google Now[?] は生活の中において、必要な情報を聞く前に教えてくれる技術である。例えば、夜遅くまで外にいるとき、ユーザがスマホに聞くことなく終電の時間を教えてくれる機能がある。これはスマートフォンの GPS 機能と、現在時刻、交通機関のダイヤを参照した上で通知を与えている。他にも、ウェアラブルデバイスが注目されている。fitbit[?] は腕時計式のウェアラブルデバイスである。アプリをインストールすると、デバイスから取得した歩行数や心拍数、睡眠時間、食事、消費カロリーなどのデータを閲覧できる。他にも、車にカメラを取り付けることで道路上の白線の掠れを検知し、塗り直すべき白線の箇所を取得する研究 [?] がある。これによって、今までは別途調査が必要であった道路の白線の掠れている場所の検知が簡単になった。このように、IoT 製品· サービスは様々な利益を我々に与えてくれている。そしてこれらの IoT 製品· サービスは全て取得した IoT データから我々に有益な情報を与えてくれているのだ。この元データなしに IoT の製品· サービスは決して生まれない。そこでこの IoT データの流動性を高めるため、IoT データ市場というものが近年、考えられている。その市場では IoT データを事業者間で売買できるようになっていて、取引の際の手数料をこの市場を管理する管理者へ払うようになっている。他にも、この市場に参加する際や、参加し続ける際に管理者へ払うようになっている制度も存在する。このように一定の仲介手数料は存在するものの、IoT データをより簡単に調達できるようになる IoT データ市場は、買い手にとって利益をもたらしてくれるものである。またこの IoT データ市場は売り手にとっても、今まで自社でしか活用用途のなかったデータを販売することが可能になる点で、利益を得られる。このように、IoT データ市場は買い手と売り手の双方にとって利益を享受することのできるものであるため、これから IoT 市場全体の成長に伴って出現· 発展していくものと考えられている。

1.2 IoT データ市場に関する問題

この便利な IoT 製品· サービスを支える IoT データの元となり得る IoT データ市場であるが、今の状況は中央集権的なサービスである。一般に、中央集権的な管理者がいるサービスと管理者のいないサービスが存在

し、各々で良い点と悪い点が存在する。管理者のいるサービスの良い点は、秩序を整えることが容易なこと、サービスのシステムがシンプルに保てることである。例えばサービスの上で倫理的に悪いことをしているユーザを発見したとする。この際、管理者がそのユーザのアカウントを凍結するなど、秩序を乱す原因となるものを素早く取り除くことができる。また、新しい便利な機能を追加する時に、管理者がスピード感を持ってデプロイできる。自分以外に合意を取るプロセスが不必要であるためであり、これも秩序が整っていることの証左である。そして管理者のいるサービスは、サービスを提供する側とサービスを消費する側ではっきりと役割が分かれており、一般的なサービス形態であることからノウハウも溜まっている。よってサービスのシステムをシンプルに保つことができるのだ。一方、管理者がいないサービスの良い点は、そのサービスの透明性が担保されていることである。管理者がいるサービスにおいて、ユーザに対して不当に悪い扱いをしたとしても、その事実が表出しない。例えばオンラインショッピングサービスの上で倫理的にも法律的にも正しく、商売を行っていたユーザがいるとする。だがそのユーザの商売が、管理者も同じショッピングサービス場で行なっている商売と同じであり、管理者の利益を逼迫していたとする。この時、管理者は悪い行いをしたユーザと同じようにそのユーザのアカウントを凍結することができる。そしてこの事実を訴えたとしても、技術的にはアカウント凍結を覆すことはできない。しかし、管理者のいないサービスがこの事態を引き起こすことはない。なぜなら全てのサービス利用者が対等な立場でサービスを利用するため、特定の利用者が大きな力を得ることがないためである。

さて、今回注目する IoT データ市場は管理者のいるサービスといないサービスのどちらで運営されるべきであろうか。IoT データ市場は参加者にとって商機の存在するサービスであり、サービスを使うことに対して必死になる動機が十分にある。このように参加者が必死になるサービスにおいて特定の管理者がいた場合、この管理者の持つ権限は巨大なものとなる。例えば A 社と B 社がライバル関係であり、どちらも管理者のいる同じ IoT データ市場で商売をしていたとする。この時、A 社は B 社に不利になるような、あるいは A 社にとって有利になるような仕様の改変を求める可能性は十分にある。更に管理者はその改変が出来てしまうので、仕様の改変を行う代わりに管理者が A 社に対して見返りを求めることが可能になる。この見返りを求められるような力こそが、管理者の持つ巨大な力の正体である。そしてこのような状態は市場全体に対して良い影響を与えない。自社のコストカットによる値段の抑制を行う、あるいは他者との差別化を図れるデータ収集を行って商機を図るといった正しい競争原理が働かなくなる。権力者に媚びることが利益を上げる最善の手法となってしまうためだ。更に IoT データ市場は巨大な市場となりうるため、その権力者は更に大きな力を持ち、見返り目的に行動しやすくなる。このような危険性は、市場という公正で透明な取引が必要とされる場所において致命的である。ここから分かるように、市場は公正で透明な取引の担保が必要であるのだ。一方で、IoT データ市場に中央集権の管理者が存在することの利点を考えてみるが、市場はこれから多くの新機能を必要とされる種類のサービスではないだろう。また、倫理的に悪いことをしているユーザに関しても特定の団体が排除するべきものではなく、国や参加者の総意によって排除されるべきものである。特定の管理者がユーザを排除できることは市場においては逆にリスクである。

以上のように考え、市場のようなお金の絡むシステムに関しては管理者がいないほうが望ましい。しかし、現状は管理者の存在する IoT データマーケットしか存在しない。本研究ではこれが問題であると考える。

1.3　目的とアプローチ

本研究の先にある最終の目的は、利用者にとって自由の割合と支配の割合が最も理想的な塩梅で、市場の統治が徹頭徹尾透明な IoT データ市場を構築することである。具体的に述べると、最終目的にある市場は利用

者の意思や希望と関係なく、技術的に仕方なく存在していた IoT データ市場の巨大な管理者という存在を利用者にとって必要な分のルールや管理団体よって透明な形で提供するものである。そこで本研究では現実世界の市場と同じように、一つの組織にも囚われない、ルールのみによって縛られるオープンな IoT データ市場を作ることを目的とする。そして本研究の先に、利用者にとって必要なルールや必要とされる管理団体が存在する IoT データ市場が構築される取り組みが存在するべきである。この手法を取る理由は、今までの手法を発展させてより管理団体の権限を減らすことは難しいが、本研究で提示する手法を発展させて管理団体やルールを作ることは可能であるためである。そして本研究のように中央集権を無くす際、管理者のいない中での合意アルゴリズムが必要となるが、これにはブロックチェーンを使用する。また、データの買い手と売り手の間でのデータ通信が必要となるが、これにはメッセージングシステムを使用する。この二つを統合させ、IoT データ市場を作り出すことが本研究のアプローチである。

1.4　本論文の構成

本論文は本章を含めて 8 章からなる。本章では IoT が我々の生活の役に立っていることと、そのためにはデータが不可欠でその市場が誕生していること、しかしそこには管理者がいるという問題点が存在することを示した。また、それに対する目的とアプローチを述べた。2 章ではこれをさらに詳細に、技術的な観点も含めて論じる。3 章ではブロックチェーン技術について簡単に述べ、今回使用する Ethereum やオフチェーン技術について触れる。4 章では提案する IoT データ市場の機能要件およびそのプラットフォーム上での取引の流れを述べる。5 章では提案する市場に関して、設計と実装を述べる。6 章では提案する市場に関して、トランザクション流通量などの定量評価を行う。7 章では今後の展望について、ブロックチェーン技術の観点と社会的な観点から論じる。8 章では本論文のまとめを述べる。

第 2 章

背景と問題意識

この章では、本研究における背景と問題意識について詳細に述べる。

2.1 背景

最初に、本研究の背景について述べる。

2.1.1 IoT

IoT とは、物理空間の様々なモノがネットワークに繋がり、そのデータに基づいて組織の意思や他のモノの動きが決定される世界の概念を表す言葉である。特にこの一連の流れの際、人間が意図的にデータ入力をしたりデータ送信をしたりする必要がなく、これらをモノが自発的に人間にとってはシームレスに行うことを IoT という言葉で表す。そしてこの IoT は我々の生活に大きな恩恵をもたらしている。例えば既に販売されているサービスとして存在するものとして、道路事業者や交通事業者向けにその会社の自動車の GPS 情報を取得し、交通情報を提示するものがある。[1] これは、道路事業者が利用者に対する利便性の向上を、交通事業者が業務の効率化を測れるようにするものである。また、自宅の外に温度センサ取り付けることでピンポイントで温度や湿度が取得でき、その情報をスマートフォンでスマートフォンから閲覧できる製品がある。[**?**] これにより、屋外に出ることなく手元のデバイスですぐ外の気温を確認でき、例えば屋内で今日の服装を決定することができる。このように、我々は IoT によって様々な利益を得ている。

この便利な IoT であるが、これの思想に基づいてサービスやアプリケーションを作り上げるには、コストのかかる工程が大きく分けて 3 つ存在する。1 つ目は Sensing、情報を取得する必要がある。交通情報の例では、各事業者の車に GPS を設置する部分がこれに当たる。また、もしある交通事業者が直近に通っていない交通区間があったとすると、その区間の交通情報を取得することはできない。温度計センサの例では、自分の家のすぐ外に温度計を設置する部分がこれに当たる。2 つ目は Processing、情報を処理する必要がある。交通情報の例では、GPS から取得した位置情報があまり変わっていないようであればそこが渋滞している可能性があると判断することがこれに当たる。温度計の例では、特定の温度範囲を逸脱した場合、スマートフォンへ通知を送るようになっている部分がこれに当たる。3 つ目は Actuation、情報を活かして行動する必要がある。交通情報の例では、渋滞情報を地図上にマッピングしてわかりやすく提示することがこれに当たる。温度計の例では、スマートフォンや PC 上に温度を表示することがこれに当たる。なお、ここで挙げた二つの例ではどちらもディスプレイに表示することが Actuation に当たるが、他にも「工場内で温度上昇を検知した場合、工場

内の生産機器の稼働率を下げる」ということを自動で行うこともこの Actuation に当たる。以上の流れは一般に SPA(Sensing、Processing、Actuation の略) と称され、これらを経て IoT の様々な製品やサービスは構築される。

2.1.2 IoT データ市場

ところで、現状は SPA の処理を全て一つの主体が行う必要がある。これら全てで IoT サービスが出来上がるので、当然と言えば当然だ。しかし近年、これは IoT データを売買できるプラットフォームである「IoT データ市場」と呼ばれるものが出現している。その中の一つが EverySense[3] だ。EverySense は IoT データを売買できるプラットフォームである。この IoT データ市場について、先の交通情報の例を使い、様々な観点から考えてみよう。最初に、IoT データの買い手の視点に立つ。同じ道路を走る車にいくつもの GPS センサを取り付ける必要は、企業間の垣根を取り払えば存在しない。同じ道路に同一事業者の車がいないので、その区間の交通情報を取得するためにセンサを取り付ける必要があるのだ。もし他の会社の車の GPS 情報を買い、取得することが出来れば、わざわざ GPS センサを取り付ける必要はない。更に、先ほどは自社の車が通っていない交通区間についての情報を取得することはできなかったが、情報を買うことができれば通っていない道の交通情報も分かる。次に、IoT データの売り手の視点に立つ。今までは GPS センサを取り付けることは自社の為のみであった。したがって、GPS センサの代金や取り付けの工事費は全て自社のコストとなり、そのコストは顧客の払った売り上げから賄っていた。しかし GPS センサのデータが売れることが分かれば、このコストの一部はデータの買い手が負担することになり、価格面で顧客サービス向上につながる。最後に、全体を俯瞰する観点に立つ。同じ時刻に同じ場所を走行する別事業者の車両が 1 台ずつ、計 2 台が存在していたとする。片方の会社はもう片方の会社から車両データを買えば良いので、IoT データ市場の出現によって無駄な GPS センサが 1 台減ることとなる。更に、IoT 化を進める上で不可欠なセンサが物理空間に増える可能性を秘めているのだ。データの売り手がデータ取得の費用が全て既存の顧客が払った売り上げから賄うわけではないと分かった場合、更に多くのセンサを車両に取り付ける可能性がある。この時、世の中全体で使える IoT センサ量は増加し、世の中全体の IoT 化が今までより容易に進むようになる。このように、様々なステークホルダに利益をもたらし得るのがこの IoT データ市場である。

2.1.3 ブロックチェーン技術

ブロックチェーン技術の詳細については後の 3 章にて述べるが、ここではこの技術の背景と概要について述べる。詳細な理由については後述するが、IoT データ市場は管理主体が存在しないほうが望ましい。そして管理主体のいない市場を作る際は、その市場の金の流れについて全員が合意に達する必要がある。この合意に達するためのアルゴリズムがブロックチェーン技術である。合意アルゴリズムに関する研究は、現在最も有名な Bitcoin[4] の開発以前も行われてきた。完全に管理主体の存在しない研究として挙げられる'b-money'[5] では、参加者の全員が受け取れる単一の歴史を示す元帳が必要であるとした。これは現在の Bitcoin をはじめとするブロックチェーンのアイデアの中心となるものである。さらに、計算問題によって金を創造するという現在のブロックチェーンに使われているアイデアもこの論文にて導入されたが、提案が不十分であったため実装がなされなかった。これらのアイデアを Proof Of Work という具体的な手法で具現化し実装可能となり、作られたのが Bitcoin であり、ここで使われている技術や後に更に考案された技術が総称されてブロックチェーン技術と呼ばれている。現在ではチューリング完全で様々な暗号通貨の基軸暗号通貨プラットフォーム

として使われている Ethereum[6] やギャンブルのチップとして使われる Augur[7], 半中央集権的な Ripple[8] などもこのブロックチェーン技術によって存在している。

2.2 IoT データ市場に関する問題

IoT データ市場は前述の背景を経て作られることとなったが、ここには大きな問題点が存在する。ここでは、政治的な問題点と技術的な問題点の 2 点に分けてその問題点を説明する。

2.2.1 政治的な問題

最初に政治的な問題点について説明する。政治的な問題、それは市場に単一の管理者が存在することだ。そして単一の管理者は、その管理者の一存で市場を動かせるので、巨大な力を持つ。そしてこの巨大な力は市場において 2 つの問題を孕んでいる。1 つ目は市場の管理者が市場の前提条件を簡単に覆せること。2 つ目は公正な市場の担保が難しくなること。1 つ目、市場の管理者が市場の前提条件を簡単に覆せることについて考察する。市場の管理者のビジネスモデルの代表的なものの一例としては、市場参加者がデータの売買をする際、プラットフォーム提供料として販売手数料を徴収する方法である。この販売手数料が例えば 10% で設定されているとする。すると、データの売り手は「10% の販売手数料であれば例えば A 円で販売し、このデータが B セット売れると考えられるので A × B 円が売り上げになる。そのためには 円のセンサを取り付けることによって最大の利益が得られる。」という計画で販売計画を立てる。この販売計画の根底にあるものは「10% の販売手数料」という前提である。市場の管理者は他の誰の同意を得ることなしに、この 10% という値を 30% へ値上げすることが出来るのだ。勿論、この値上げについては基本契約書での取り決めや、この市場に参加するまでのやりとりによっては参加者が法的に拒否することは可能である。但し、法的に解決するには長い期間や訴訟のための費用がかかる上、今回の IoT データ市場において司法がどのような判断を下すかは不明瞭である。換言すると、管理者の存在が IoT 市場において本格的に商売をしようとする事業者にとって信用する以外に方法のない存在であるのである。つまり、この管理者が全ての善意の IoT 市場の参加者にとって「正しく」機能する必要があるが、このことについて確実に担保する術は存在しない。以上、市場の前提条件を覆せる可能性について言及した。2 つ目、管理者の存在によって、公正な市場の担保が難しいという点について考察する。一般には、公正な市場を守るために、以下の流れが存在する。以下については、2010 年 11 月 17 日時点で、証券取引等監視委員会事務局の特別調査課長であった目黒克幸氏のスライド [9] を参照した。

1. 立法権を持つ国会が公正な市場を実現するための法整備を行う。
2. 金融庁の証券取引等監視委員会が、法律にもとづいて実際の市場の監視· 調査を行い、問題があれば告発する。
3. 告発された内容に基づいて地方検察庁が起訴を行い、裁判所によって裁判が行われる。

もし IoT 市場においてもこの流れを踏襲する場合、この流れにおける全てのステップに、今回の IoT 市場は市場の管理者が関連することとなる。我が国では立法権、行政権、司法権と独立した 3 権の行う権利行使に一つの市場管理者が関与するのだ。これで公正な取引が担保される可能性は大きく減る。例えば市場の管理者にとって、ビジネス的に敵対する事業者が市場に参入しようとしたとする。あるいは、市場において有力な参加者がある事業者を排除しようと、市場の管理者に何らかの方法で参入しないように圧力を加えたとする。これに応じた管理者は、新規参入しようとした特定の企業を排除するような制約を参加する企業に課すことが出来

る。もしこのようなことを行政が行おうとし、それが立法府の作った法律に違反しているようであれば司法がこれを許すことはない。しかし、三権が一つの管理者に集中しているこの IoT 市場は、この参加制約を簡単に作り出せてしまう。なぜなら市場の管理者内での結論というブラックボックスによってルールが作られ、そのルールに基づいた実効支配が行われ、その支配が正しいものであるか否かが決められるからである。これでは公正な市場を担保することは難しい。以上、管理者の存在による IoT データ市場の政治的な問題点を大きく分けて二つの観点から述べた。

2.2.2 技術的な問題

可用性とセキュリティの観点から、管理者の存在する IoT 市場の問題点について述べる。最初に、可溶性の観点から述べる。IoT 市場は一瞬であろうと市場取引やデータ送信が止まる事は望ましくない。だが、特定の一つの管理者のプラットフォーム上で動く以上、稼働率 100% を担保することは難しい。例えば、クラウドサービスとして有名な AWS(Amazon Web Service) の EC2 などの稼働率に関する SLA(Service Level Agreement) は最高で 99.99% である。この 99.99% を割り込んだ場合、サービスクレジット率の 10% がこれから AWS の製品を使う上で使える金となる。仮に稼働率をこの SLA の 99.99% とした時、1 年間で AWS が稼働していない時間は 52.56 分である。オンプレミス環境と比べて可用性に比較的信頼が置かれているクラウドでさえ、1 年単位で考えると 1 時間弱程度のダウンタイムは仕方がないと AWS は考えている。この 1 時間弱の間にどれほどの裁かれるべきデータ送信が滞るのか。IoT データは逐次飛んでくるものであるので、この時間の間に大量のデータ送信が滞ってしまうことは想像に難くない。また今回はクラウドを想定したが、管理者がクラウドサービスを使い可用性を 100% に限りなく近づけるような努力がなされているかどうかは市場の参加者からはチェックすることができない。このように逐次的に大量のデータが流れ、それが止まってしまうと大きな問題の起こる IoT データ市場においては、単一の管理主体がその市場全体を管理することは望ましくない。次に、セキュリティの観点から述べる。当然、管理者であっても買っていないデータを勝手に閲覧することは許されない契約を市場の参加者と管理者間で結ぶだろう。ただ、それであっても管理者が売買データを見ることが可能である。また、どの企業がどのようなデータを買っているかについても、管理者は全て見ることができる。これは管理者が悪意を持っていない前提ならば問題のない話であるが、悪意を持っていた場合は参加している事業者の IoT 戦略が全て管理者に筒抜けであることを意味する。またセキュリティの脆弱性を突かれた場合、取引データが管理者のデータベースから抜かれた時には事業者間のプライバシーである取引履歴が、IoT データが抜けれた時には販売価値のある IoT データがそれぞれ不特定多数の人間によって見られる可能性を含んでいる。このように、単一の管理者が多くの流出を避けるデータを持つことはなるべくあってはならない。以上 2 点について、単一管理者の存在する IoT 市場の問題点について技術的観点から述べた。

2.2.3 まとめ

本章では IoT が我々の生活の役に立っていることと、そのためにはデータが不可欠でその市場が誕生していることを述べた。しかしそこには管理者がいるという政治的な、技術的な問題点が存在することを示した。そしてこの状況を改善するために、ブロックチェーン技術というものがあることを示した。

第 3 章

ブロックチェーン技術

本章では、本研究で用いるブロックチェーン技術について述べる。最初に仕組みや説明を述べ、その技術が持つ問題点や現状の政治的な問題点について述べる。その後、ブロックチェーン技術を最大限に活用するための技術であるオフチェーン技術について述べる。そして、ブロックチェーン技術を用いた暗号通貨の代表的なものについて説明を行う。最後に、本章のまとめを行う。

3.1　仕組み

本節では、ブロックチェーンの仕組みについて述べる。また詳しくは後の**??**項にて後述するが、ブロックチェーンにはいくつかの種類が存在する。ここでは特に指定のない限り、中央集権的な機関の存在しない、パブリック型ブロックチェーンについての説明を行う。

3.1.1　P2P 通信

ブロックチェーンは P2P 通信によって行われる。この P2P 通信とは、対等の端末間で行われる通信のことである。通常のネットワークサービスは、クライアント・サーバ型と呼ばれる通信機能によって運営されている。サーバ側ではサービス運営者がサービスや機能を提供するアクセス可能なコンピュータであるサーバを設置し、このサーバ上でサービスが運営される。例えば慶應義塾大学湘南藤沢キャンパスにて使われている学事システムの SFC-SFS では、履修可能単位の閲覧機能、履修単位申告機能、履修者が過剰になった時の選抜機能などを提供している。この機能を運営しているコンピュータを一般に、サーバと呼ぶ。これに対し、利用者はクライアント側となる。クライアントとは、サーバに対してそれが持つ機能を使わせてもらうためのリクエストを送るコンピュータのことである、SFC-SFS の例では、学生がサービスにアクセスするために使うスマートフォンや PC などがこれに当たる。つまり、サービスの提供者側であるサーバと消費者側であるクライアントで役割が分かれていることが特徴だ。このような形で通常のネットワークサービスは運営されるが、P2P サービスはこれと異なる。P2P サービスでは、通信する端末間の関係が対等である。つまり通信する双方が同じ機能を持ち、相手へサービスを提供する一方で、相手からサービスを受けているという状況が発生しているのだ。P2P サービスの例として、インターネット回線を使った通話アプリが存在する。P2P の通話サービスの場合、A の端末と B の端末で通話を行なっている時、この通話アプリを提供している会社は二人の会話中の通信について関与していない。双方ともが自分の音声を相手へ提供する機能と相手の音声を受け取る機能を持つ、つまりクライアント・サーバ型の両方の機能を双方が持っているのだ。以上が P2P 通信の特

徴である。ブロックチェーン技術は中央集権を持たない環境での分散台帳技術であるが、これには P2P 通信が使われている。したがって、全ての参加者がサービスの使用者としての役割のみならず、サービスの提供者としての役割も持っているのだ。

3.1.2 デジタル署名とアドレス

デジタル署名とは、あるメッセージが署名した人によって作られたかを検証する仕組みである。ここでは「A が自分の 3BTC(Bitcoin) を使いたい」と主張する時に、それが本当に A の発言であるかを担保するのがこのデジタル署名の役割である。代表的なブロックチェーンである Bitcoin や Ethereum では ECDSA(楕円曲線 DSA, Elliptic Curve Digital Signature Algorithm) を利用しており、これについて説明を述べる。(!!! 後に参照する図を挿入のこと！)

$$y^2 = x^3 + ax + b \tag{3.1}$$

以上が楕円曲線について一般的に表される式である。この中でも Bitcoin や Ethereum が用いる規格である secp256k1 曲線は a=0, b=7 であるため、以上の方程式は

$$y^2 = x^3 + 7 \tag{3.2}$$

上記のように表される。また楕円曲線の加算の定義として、点 A と点 B を加算することを考える。この時、加算後の座標は点 A と点 B とを通る直線のもう一つの交点の x 軸に関して対称移動させた点である。したがって点 A と点 B が同一座標の点 G であるとき、その接線と楕円曲線との交点を x 軸に関して対称移動させた点は G + G = 2G となる。secp256k1 は G のベースポイントを定めており、そこから秘密鍵 r を掛け合わせた rG が公開鍵となる。この時、楕円曲線上の離散対数問題によって秘密鍵から公開鍵を導出することは容易であるが、逆の公開鍵から秘密鍵を導出することは難しいことが知られている。この秘密鍵を使い、「自分の Bitcoin を使いたい」と主張することによって、利用者は自分の Bitcoin を使用する事が可能となる。この主張を行う際の署名値は以下の式によって導かれる。

$$S = \frac{h + kR}{q}(modp) \tag{3.3}$$

q：一回のみ使われる乱数 ($1 \leq q \leq 2^256 - 2^32 - 977$)
h：取引情報のハッシュ値
k：送信者の秘密鍵
R:一時的な公開鍵の x 座標
p:楕円曲線の x 座標がこれより大きくならないための値で素数

そしてこれらの値のうち、S と R が署名となり、ブロックチェーン上で周知される。この主張が本人のみ知り得る秘密鍵を使って行われたものかを確認する際は、以下の式を使って検証する。

$$Q = \frac{hG}{S} + \frac{RK}{S}(modp) \tag{3.4}$$

S, Q, R：送信者から受け取った署名
h：取引情報のハッシュ値
G：secp256k1 のベースポイント
K：送信者の公開鍵

p:楕円曲線の x 座標がこれより大きくならないための値で素数

以上の式において、Q の x 座標が送信者が送信した R の座標と一致する時、この署名は正しいものであると検証される。また、この公開鍵を Bitcoin では HASH160 · Ethereum では Keccak256 ベースのハッシュ関数によってそれぞれハッシュ化し、Ethereum ではそのハッシュ値の末尾 20 バイトを抜き出したものがアドレスとして使われる。そして「そのアドレスに対して、3BTC を送金する」と主張できるようになり、このアドレスの管理者 (つまり元の公開鍵や秘密鍵を持っている者) がその後、「ここで受け取った 3BTC を使う」と主張できるようになるのだ。(!!!後に参照する図を挿入のこと！)

3.1.3 トランザクション

ここでは、Bitcoin を例にとトランザクションについて説明を行う。ブロックチェーン上の記録は、全てトランザクションという単位毎に格納される。つまり、「A が B に 3BTC を渡した」という記録が一つのトランザクションに格納されるということだ。このトランザクションはインプット部分とアウトプット部分、そしてその他の部分が存在する。インプットには、当該トランザクションのトークンの出所が存在している。例えば、「A が 3BTC を使う」と申し出たとしよう。この時、ネットワーク全体が「A は 3 BTC 以上持っている」ということが分からないと、A が 3BTC 使うという行為は認められない。ここで「3BTC 以上持っている」ということは、換言すると「3BTC 以上を誰かから送金された過去があり、その BTC は未だに使用されていない」ということである。この「未だに使用されていなく、その所有者が未だ使える状態」のトランザクションのことを Bitcoin では UTXO(Unspent Transaction Output) と呼ぶ。この UTXO を使おうとする時は、トランザクションのインプットに UTXO の存在する場所を明示することで、UTXO を使う事ができる。つまり、インプットはトランザクションの送金における払い手に当たる情報が入る部分と言える。次に、アウトプットについて説明する。アウトプットはトランザクションの送金における受け取り手に当たる情報が入る部分である。前項で述べたように、受け取り手の情報はアドレスによって表される。したがって、「A が B に 3BTC を渡した」あとで未だに B がこれを使っていない状態の時、このトランザクションのアウトプットの署名欄には Bitcoin アドレスが存在している。その後、「B が C に 3BTC を渡した」とすると、先ほど Bitcoin アドレスが書かれていた署名欄には Bitcoin アドレスの素となった公開鍵とこの公開鍵に対応する署名値が代入される。つまり、あるアウトプットが UTXO であるか否かの判断はこの署名欄に Bitcoin アドレスがあるか公開鍵と署名値が存在するかの違いによって行われる。このようにしてトランザクションは管理される。

3.1.4 ブロックとマイニング

ここではブロックとマイニングについて説明を行う。ブロックチェーン技術は情報の記録を単一の歴史を共有することによって、参加者の合意できる台帳管理を行おうとする技術である。この単一の歴史を刻む歴史書の 1 ページが 1 ブロックに当たる。現在の Bitcoin では 10 分に 1 回、Ethereum では 15 秒に 1 回、それぞれのペースで新しいブロックが生成される。このブロックにはトランザクションが 0 個以上含まれており、その処理内容が単一の歴史として刻まれる。そしてブロックチェーン技術はこの方式によって、二重支払い問題を解決している。A が 3BTC を持っている時、同時に「A が B に 3BTC を払う」と「A が C に 3BTC 払う」というトランザクションを発行しようとしたとする。しかし、ブロックが生成される際にのみ送金の処理は行われるため、ネットワークの遅延等の影響によって二つのトランザクションが承認されることはあり得ない。また、各々のブロックはヘッダに前のブロック情報をまとめたハッシュ値を持っており、どのブロッ

クの次に繋げられたブロックであるかを明示している。このブロックがチェーンのように何個も連なることによって、ブロックチェーンという分散管理台帳が形成されていく。そしてこのブロックが生成される際に行われることが、マイニングと呼ばれる行為である。そしてブロックの生成を行おうとする者をマイナーと呼ぶ。ブロック情報をまとめたハッシュ値の中には、ナンスと呼ばれるブロックのマイナーが付加する 32bit の数値が存在する。このナンスを含めたハッシュ値が、一定の数だけ頭に 0 を持つようにすることによって、そのブロックは正当なブロックであると承認されるようになっている。例えば、Bitcoin のメインネットワークにおいて 553582 番目に生成されたブロックの情報について見てみる。以下は Bitcoin の現在の統計情報などを提供している https://www.blockchain.com/ja/ から取得した情報である。すると、ハッシュ値は「0000000000000000001b0218ca2b54e9809b5d948864c5bd1e657e5aa09f438f」となっている。そしてこの時のナンスの値は「4142738813」となっている。ブロックの持つトランザクションのハッシュ値などの情報に、このナンス値を足したところ、このように頭にいくつもの 0 がつくハッシュ値を見つけ出せたのである。この時この様々なナンスの値を取り付け、0 が頭に一定数以上つくハッシュ値を見つけ出す行為について、マイニングと呼ぶ。そしてこのマイニングという行為によって、ブロックチェーンにおける改竄可能性を防いでいるのだ。現在 (2018-12-13 00:46:30)、マイニングは 16 進数において頭の 18 文字に 0 が続く場合、ブロックが生成されるようになっている。つまり、最新より一つ前のブロックのハッシュ値をブロックに含めて $16^{1}8$ 回の演算を行うことで最新のブロックを無効にでき、自分の思い通りのブロックを提出する事ができる。しかし、2 つ前のブロックを変更しようとするときはどうだろう。Bitcoin の各参加者は、ブロックがもっとも長く連なったブロックチェーンを信用するように設計されている。つまり 2 つ前を変更するには、新しく 2 つ分のブロックを生成しなくてはならないのだ。この時に必要な計算量は $16^{(}18 \times 2)$ となり、難易度は格段に上昇する。これがさらに 3 つ前、4 つ前.. となっていくと、事実上変更は不可能となる。よって 3BTC を払い、その対価としてのサービスを受けたのちにその支払った BTC を取り返すために新しいブロックを作り直す行為は、ブロックが一定数以上深くなった場合においては不可能である。つまり理論的にブロックの改竄は可能であるが、それは実際にはそれを行うことは極めて難しいということである。これがブロックチェーンの参加者が公開台帳を信用する理由であり、改竄耐性を持つという理由である。そしてマイナーがこのマイニングを行う動機はマイニングを成功した時に成功報酬がもらえることである。この成功報酬はブロック高によって決められており、最初は 50BTC で始まり、2018 年 12 月現在では 12.5BTC である。このようにしてブロックは生成され、ブロックチェーンは管理される。

3.2 問題点

ここでは、前半の 2 項でブロックチェーン技術に関する固有の問題点を述べる。その後、後半の 2 項で現在のブロックチェーン事情に関する問題点を述べる。

3.2.1 スケーラビリティ

ここではブロックチェーンで管理することによる、トランザクション処理数に関するスケーラビリティの少なさについて述べる。Bitcoin では 1 ブロックに含める事が可能なデータ量は 1MB となっている。このデータ量の中にトランザクションに関するデータを含めなくてはならないため、1 ブロックで処理できるトランザクションの数は限られている。そしてブロックの生成ペースは平均で 10 分に 1 回になるように調整されるため、処理できるトランザクションの数は時間当たりで一定であると言える。つまり Bitcoin を 100 人が使おう

が、100 万人が使おうが、同じだけの処理能力しか存在しないのである。例えば Web サービスであれば、受け付けるリクエストに対する処理能力を上げる方法はたくさんある。Web サーバの台数を増やし、ロードバランサで負荷を振り分ける。DB を分散処理させ、データモデリングを見直してよりレスポンスの速い DB にする。JavaScript を後から読むようにし、思い画像は最初のトップページ範囲を表示してから順に Ajax で取得する。など、様々な方策を打てる。しかし、ブロックチェーンは時間をかけてマイニングを行い、複数のチェーンが同時並行で存在していくことを防いでいる。もし 3 秒に 1 回のペースでブロックが生成される場合、同じ長さのチェーンが大量にできてしまい、どれが本当の台帳とみなして良いかわからなくなる。また、ブロックには一定のサイズしか入らないことで最新のブロックが変更された時の巻き戻されるトランザクションの数を減らしている。もし大量のトランザクションを 1 つのブロックに入れた場合、それはブロックを広報する際の遅延が生じる。この時、最新のブロックからマイニングを行いたいマイナーは、ブロックの遅延によって不利を受ける。これが続いた場合、マイナーが少なくなり、マイナーの一極集中を招く恐れがある。このことによる弊害については 3.2.4 にて後述する。つまり、仕組みそのものに固有のスケーラブルになり得ない要素が含まれているのだ。勿論、マイニングには参加者が多い方がマシンパワーが大きいので、変更されにくいブロックを生成する事が可能である。しかしながら、それは改竄耐性が上がるのみであり、トランザクションの処理能力は上がらない。どれだけ沢山の参加者が増えたとしても、それはセキュリティ性を高めるために使われてスループットを犠牲にしている、ここにブロックチェーンの大きな問題点の一つが存在する。

3.2.2　手数料とマイクロペイメント

全世界では大量のトランザクションが生成されている。従って、生成された全てのトランザクションが直ぐに最新のブロックに入るとは限らない。この時、早くマイナーにブロックへ入れて貰うため、トランザクションの送信元は手数料を設定することができる。この手数料はマイナーが得る成功報酬にプラスして、マイナーへと渡る。よって、より多くの手数料を指定した方が早くトランザクションが処理される可能性が高まるのだ。またブロックサイズの上限が決まっているため、トランザクションの大きさが大きい程、多くの手数料がマイナーへのインセンティブに必要となる。1MB の内 500KB を使うトランザクションを処理して 0.01BTC の手数料のトランザクションと、1MB の内 50KB を使うトランザクションを処理して 0.01BTC の手数料のトランザクションとでは残りのブロックに入れられるトランザクションの大きさが変わってくるためである。残りでより大きなトランザクションを捌ける方が、より多くの手数料を獲得できるためである。しかしこの仕組みは、小さな買い物に使うためには適していない。手数料はトランザクションベースで決まるため、少額の支払いであれば手数料が少なくて済むという性質のものではないためである。つまり、Bitcoin で 10 円にあたる飴を買う際も 1000 万円にあたる高級車を買う際も、同じ速度で処理してもらうには同じ手数料が必要なのだ。よってここまで記したのブロックチェーン技術では、マイクロペイメント (少額取引) には適さないという問題点がある。

3.2.3　マイナーの一極集中

ブロックはブロックチェーンにおける歴史書の 1 ページであり、マイナーはそのブロックを承認する役割を持っている。即ち、マイナーは取引の歴史の承認者、ブロックチェーンの管理者と換言できる。そしてそもそも、ブロックチェーンは完全に分散された記録システムであった。中央集権な機関が存在しないので、その完全に透明性が保たれたプラットフォームを信用する人が参加するものとして作られた。しかしながらこの

前提を覆すようなことが Bitcoin では起こったと、Bitcoin のコア開発者で Blockstream の共同設立者である Pieter Wuille 氏は言う。(http://nonem.hatenablog.com/entry/2017/10/14/182226) マイナーの中にはマイニングプールと呼ばれるものを作り、ブロックを生成している集団がある。彼らは自宅の PC などでマイニングをしたいものの、成功する確率が少ないので大勢でマイニングを行っている。そして例えば各人の計算能力が同じ 100 人で 12.5BTC を採掘した場合は、1 人あたり 0.125BTC ずつ分配する。このいくつかのマイニングプール間で、生成したブロックを送信する前にブロックヘッダのみを共有しているようなのだ。これにより、情報を共有するマイニングプールは他のマイナーよりも早くマイニングに取りかかることができる。これは計算能力を信用しているのではなく、共有の相手のマイニングプールを社会的に信用していることであり、これは Bitcoin の基本理念に反する。それと同時に、これらのマイニングプール間でのマイニングが有利となり、他のマイナーがマイニングに参加する際に不利となってしまう。ここで、これからマイニング参加するにはこのマイニングプールに所属する方法が一番良い方法となる可能性がある。その際マイナーは歴史の承認者であるため、一極集中するとその間でメインの Bitcoin とは別の取り決めを作ってしまい、それが Bitcoin のルールとなってしまう可能性がある。この時、本来の目的と離れて Bitcoin に中央集権的な機関が存在してしまう可能性がある。以上の理由から、マイナーの一極集中は望ましいことではなく、現在の Bitcoin を取り巻く状況の問題点の一つである。

3.2.4 現在の OSS ブロックチェーンの運営

”Decentralization in crypto is a myth. It is a system more centralized than North Korea: miners are centralized, exchanges are centralized, developers are centralized dictators” 「暗号通貨が中央集権的でないと言うのは神話である。そのシステムは北朝鮮よりも中央集権的である。マイナーは中央集権的で、交換所は中央集権的で、開発者は中央集権化された独裁者である。」以上の言葉はニューヨーク大学の Nouriel Roubini 教授が twitter で発言した言葉である。確かに、現在のブロックチェーンはパブリック型であってもその仕様の決められ方は決して民主的なものではない。Ethereum は 2019 年の 1 月にハードフォークが行われることが決まったが、これは開発者会議によって決まったものだ。それ以前でも、Ethereum は The DAO 事件の時のハードフォークから中央集権的であると批判を集めてきた。The DAO という Ethereum 上で動くトークンの資金集めに 150 億円分の Ethereum が集められた。しかし、Ethereum 上で動かす solidity と言う言語のフォールバック関数の仕様に関しての見落としがあり、3 分の 1 が攻撃者によって抜き取られてしまった。その際、Ethereum コミュニティは歴史書であるブロックチェーンの巻き戻しを行い、攻撃者が利益を得ることを阻止した。このような対策が一プロジェクトのバグに対して行われること自体が、中央集権的であることの証左である。ここにマイナーや ETH(Ethereum の通貨単位) を持っている人間の意思は反映されていない。マイナーの件に関しては前項で述べたものがそのままこのツイートの論拠となる。このように、現在の OSS ブロックチェーンはパブリック型と謳いつつ、とても中央集権的であるという側面を持っている。

3.3 オフチェーン技術

ここではオフチェーン技術について述べる。今までのブロックチェーンに関する説明はこれに対応してオンチェーンと呼ばれることがある。オンチェーンはトランザクションの制限が厳しく、全世界で使われる通過の処理が 10 分に 1 回しか行われない。Bitcoin が始まって以来、一日のトランザクションが 45 万を超えたことはない。これは、秒間 5.2 トランザクション以上が処理されたことがない計算になる。VISA を始めとし

た決済システムは遥かに多い TPS(Transaction Per Second) を実現しており、世界中の決済を目的とするには 5TPS は明らかに少ない数字である。この制約を緩やかにするため、オフチェーン技術と総称される技術が存在している。例えば 1 曲 100BTC で楽曲を配信するサービスを考えてみる。最初に、ユーザは使いたい BTC をオンチェーン上にデポジットする。10 曲分配信サービスを受けたいと予定したすると、1000BTC をデポジットする契約をトランザクションとして広報する。その後 1 曲の配信サービスを受けるとき、支払い側は 100BTC 分を払うという署名を行ったトランザクションを受け取り側へ送る。受け取り手はこのトランザクションを確認次第、1 曲の楽曲を配信する。そして支払い側がもう 1 曲楽曲が欲しい時は今度は 200BTC を払う署名を行ったトランザクションを受け取り側へ送る。というこの繰り返しを行い、もう支払い側が要らないと思った時、このオフチェーンでのやり取りを終える。この時の実際の操作としては、受け取り側が精算する旨をトランザクションとして広報することを行う。そしてもし 6 曲の配信サービスを受けた時、600BTC が受け取り側に、400BTC が支払い側に支払われる。この時、最初と最後の 2 つのトランザクションのみオンチェーンには広報したが、実際にはオフチェーンによって 6 曲分の支払いがなされている。オンチェーンのみでは 6 つのトランザクションが必要なところを 2 つのトランザクションのみで同じ機能を提供できたということである。これがオフチェーン技術の概略であり、これに対する細かな実装はブロックチェーンの種類によって違う。各々が Bitcoin と Ethereum のオフチェーン技術の一つである、ペイメントチャネル技術と μ Raiden 技術については後の 3.5.3 項と 3.6.4 項にて行う。

3.4 Bitcoin

ここではブロックチェーン技術を使って作られた最初の実装物である Bitcoin について述べる。3.1 節で説明したことと重複することが多いが、のちの Ethereum と対比するために述べる。通貨単位は BTC である。

3.4.1 トランザクションベースの一元管理

Bitcoin は全てトランザクションベースで管理される。3.1.2 項で述べたように、自分の BTC を使いたい時はその BTC をもらった過去のトランザクションを指定する。そしてそのトランザクションを指定して BTC を使った過去がないことを承認ノードが確認 (UTXO であることを確認) した上で、その BTC は使用される。ここで、仮に 10BTC を貰ったトランザクションを使って 3BTC を使いたいとする。すると支払い側は生成されるトランザクションは 3BTC を支払い相手へ渡し、残りの 7BTC は自分のアドレスへ送るように指定することで 3BTC のみを使うことが可能となる。この工夫が一般的である理由は、トランザクションベースの一元管理であることに存在する。もし 7BTC を明示しない場合、お釣りの 7BTC がそのまま送信者のアドレスに紐づけられたままできるのであればトランザクションのサイズを減らし、トランザクション送信の際の手数料を少なくできる。お釣りのアドレスを指定することは送信相手を複数指定するためにトランザクションのサイズが大きくなることを招き、このことがトランザクション送信の際の手数料増大を招くためだ。もしアカウントベースで Bitcoin が管理されていれば、この方式を行うことに一定の負のインセンティブが存在するのだ。また、トランザクションベース管理ではこのお釣りであるという情報を秘匿する工夫も考えられている。もし同じ Bitcoin アドレスへ 7BTC を送った場合はこの 7BTC がお釣りであることが明確であり、本来は公開されるべきでない支払い情報の一部が全世界にバラされてしまう心配がある。よって、これに対する一般的対策として BIP(Bitcoin Improvement Proposal)-32 で提案された、拡張鍵生成が知られている。これはチェーンコードやインデックスの数字を用い、新しく秘密鍵· 公開鍵· Bitcoin アドレスを生成する方法の一

般的方法である。これを使うことで、同じく自分が管理しているアドレスでありながらマイナーや他の参加者からはどちらがお釣りでどちらが本来の支払いに使われたのか、あるいはどちらも別々の支払いに使われたのかが分からなくなる。このようにして、支払い情報についてなるべく公開されないような工夫が一般的に行われている。また、Bitcoin が徹頭徹尾トランザクションベースで管理されていることがこれらの振る舞いや工夫から分かる。

3.4.2 script 言語とチューリング不完全

Bitcoin の署名に関して、具体的な方法について今まで言及してこなかったが、これが script 言語と呼ばれるもので行われているという具体的プロセスをここで記す。UTXO の使用時、script 言語と呼ばれる言語によって記述されたプログラムが正を返す時、その UTXO は使用可能となる。Bitcoin の使用例としてもっとも代表的である、Bitcoin をあるアドレスからあるアドレスへ移動させる場合を考える。その際のプログラムは以下のようになる。

ソースコード 3.1: script 言語

```
<sig> <pubK> DUP HASH160 <pubKHash> EQUALVERIFY CHECKSIG
```

<sig>：秘密鍵でトランザクションに署名したもの
<pubK>：公開鍵情報
DUP：一つ前の内容をコピーする命令
HASH160：公開鍵からアドレスを BASE58 でデコードしたものを導く関数
<pubKHash>：アドレスを BASE58 でデコードしたもの
EQUALVERIFY：スタックに積まれている一つ前ともう一つ前が同じであることを確認する命令。異なればその時点でプログラム全体の返り値が False となる。
CHECKSIG：二つ前の署名値が一つ前の公開鍵情報に対して正しいか否かを判断する。

この script 言語は逆ローランド記法であり、順に命令がスタックに積まれて実行されていく。なお、この時トランザクションのアウトプットに記述される、トランザクションをロックするためのプログラムが以下である。

ソースコード 3.2: ロックを行う script 言語

```
DUP <HASH160> <pubKHash> EQUALVERIFY CHECKSIG
```

そしてインプットに記述されるトランザクションをアンロックするためのプログラムが以下である。

ソースコード 3.3: アンロックを行う script 言語

```
<sig> <pubK>
```

つまり、「アンロックのためのプログラム」と「ロックのためのスクリプト」をこの順番で続けて実行することでトランザクションへの署名が正しいかの判断は行われる。以下にその時の様子を示す。

1. <sig>がスタックに積まれる
2. <pubK>がスタックに積まれる

3. DUP によって<pubK>が複製される
4. HASH160 によって 3 番目で複製された公開鍵情報がアドレスのデコードされた状態の値になり、スタックに積まれる
5. <pubKHash>がスタックに積まれる
6. EQUALVERIFY によって 4 番目でハッシュ化されてスタックに積まれたものと 5 番目でスタックに積まれたものとを比較する。同じであった場合は実行中のプログラムが続行され、異なっていた場合は実行中のプログラムは中止する。
7. CHECKSIG によって、1 番目でスタックに積まれた署名値と 2 番目でスタックに積まれた公開鍵情報を検証し、正しいものであれば真を返し、間違っていれば偽を返す。

以上が script 言語の実行内容である。また、この script 言語の特徴として、チューリング不完全であることが挙げられる。このことにより、チューリング完全である時と比べてセキュリティホールが少なく済むことが知られている。その一方で、このスクリプト言語によりユーザが望む様々な処理が実現できるとは限らない。

3.4.3 ペイメントチャネル

Bitcoin におけるオフチェーン技術であるペイメントチャネルは、トランザクションの持つロックタイム機能とマルチシグ機能を併用して実現される。ロックタイムとは、定めた時間になった時までそのトランザクションが実行されない機能のことである。マルチシグとは、ある UTXO を使う際に複数の署名値が要求できる機能のことで、「N of M のトランザクション」などと表される。これは M 個のアドレスの内、N 個のアドレスの署名値が必要となる UTXO であるということを示す。例えば A が消費者· B が販売者とし、A から B へ 1000BTC のデポジットを最初のトランザクションとして持っておき、そこから 100BTC ずつで楽曲を買うことのできる状況を想定してみる。

1. A は A と B の署名値が必要な 2 of 2 のマルチシグのトランザクションを生成し、ブロードキャストする。同時に、2of2 のマルチシグなので Bob が音信不通になった時の保障のため、B が何も行わない場合 A に全ての BTC が戻るトランザクションをマルチシグのインプットから提出する。この際、この戻るトランザクションに関しては一定期間が過ぎた後に実行されるようにロックタイムを掛けておく。
2. A が 1 曲の楽曲を買うため、マルチシグのトランザクションに A のみが署名し、アウトプットとして A に 900BTC· B に 100BTC を支払うトランザクションを B へ送る。
3. A が更に 1 曲の楽曲を買うため、マルチシグのトランザクションに A のみが署名し、アウトプットとして A に 800BTC· B に 200BTC を支払うトランザクションを B へ送る。

この時、B は自分に 100BTC でも 200BTC でも送ることが可能な権利を持つが、通常は 200BTC を貰う方を選択する。このようにしてチェーンの外での取引が実現される。そして、A が払うという範囲においては、B の方から A へ BTC を送ることも可能である。ここでは先ほどの例のシチュエーションにプラスして、1 ヶ月ごとに抽選があり、それに当たると 50BTC が帰ってくるという場合を想定してみる。

1. A は A と B の署名値が必要な 2 of 2 のマルチシグのトランザクションを生成し、ブロードキャストする。同時に、2of2 のマルチシグなので Bob が音信不通になった時の保障のため、B が何も行わない場合 A に全ての BTC が戻るトランザクションをマルチシグのインプットから提出する。この際、この戻るトランザクションに関しては一定期間が過ぎた後に実行されるようにロックタイムを掛けておく。

2. A が 2 曲の楽曲を買うため、マルチシグのトランザクションに A のみが署名し、アウトプットとして A に 800BTC· A と B のマルチシグに 200BTC を支払うトランザクションを B へ送る。そして同時に、A と B のマルチシグに A が署名を行い、そのマルチシグから B へ送るようにする。この際、B へのトランザクションには 1 番目に生成したロックタイムより前に設定されたロックタイムを設定しておき、B のアドレスは今回のみ使われる一時的なアドレスを使用する。
3. ここで A が抽選にあたり、B は A へ 50BTC を支払おうとする。しかし A が 850BTC を持ち、B が 150BTC を持つトランザクションを作ったとしても、B が先ほどの 200BTC を貰えるトランザクションを提出しない保証はない。そこで、B は一時的に利用したアドレスの秘密鍵を A へ送る。これによって、B は A に前の契約より少ない BTC の契約に同意したことを示す。

もし B が 200BTC のトランザクションをブロードキャストした場合は、A はその後に続くトランザクションがロックタイムに到達する以前に A と B(一時的なアドレス) のマルチシグから A へ 150BTC を送るトランザクションを発行できるので、B はそのためのトランザクションをブロードキャストしない。B がブロードキャストしなければ、A に全額が渡るトランザクションはインプットが存在しなくなるためだ。このようにして Bitcoin のオフチェーンは実現される。

3.5 Ethereum

ここでは本研究で用いるブロックチェーン技術である Ethereum について述べる。3.4.1 と 3.5.1 が、3.4.2 と 3.5.3 が、3.5.3 と 3.5.4 がそれぞれの特徴に対応している。通貨単位は ETH である。

3.5.1 トランザクションベースとアカウントベースの二元管理

Ethereum はトランザクションに基き、アカウント毎に Ethereum を管理している。もちろん、Bitcoin と同様に「3ETH を持っている」=「3ETH を過去に送ってもらったことがある」という考えのもと、A が 3ETH 持つには A に 3ETH を送ったトランザクションが存在しなくてはならない。しかし、このトランザクションの結果、A の持つアドレスに 3ETH が紐づくのだ。つまり、3ETH を使う際は前のトランザクションに署名をして使うのではなく、「3ETH 使います」ということを A のアカウントの署名で提出すれば 3ETH が使えることになるのだ。これは面倒なお釣りの処理を行う必要も無くす。A が 3ETH 持っていて、B に 2ETH を送信した時、別途自分用のお釣りのアドレスを用意せずとも残りの 1ETH は自分のアドレスに紐づいているのだ。問題点としては、ハードフォークが起こった時にバージョンや ChainID を変更ことができないことが起こってしまうと、リプレイアタックの攻撃の可能性があることである。The DAO 事件を発端に、Ethereum は急遽 Ethereum と Ethereum Classic にハードフォークすることとなった。ブロックの巻き戻しを行うか否かで意見が割れ、巻き戻したほうが Ethereum、巻き戻さなかった方が Ehtereum Classic となったのだ。そしてこの事件より前から所持していた Ethereum を送金しようとすると、それと同じトランザクションを Ethereum Classic ネットワークでも送信できるようになる。よってこの際、Ethereum Classic は持っている人の意思とは無関係に Ethereum 送金と同時に Ethereum Classic も送金されてしまう可能性を持ってしまうのだ。これはトランザクションベースの一元管理では起こり得なかったことである。アカウントベースでの管理を加えたことは、複雑な処理を可能にしたと同時に、セキュリティ面では厄介な問題を引き起こす原因を作った。

3.5.2 スマートコントラクト

Ethereum のアドレスは EOA(Externally Owned Account) アドレスとコントラクトアドレスが存在する。Bitcoin のアドレスと同様に、所有者に紐付くアドレスが EOA アドレスである。EOA アドレスからは採掘や送金などを行うことができる。Bitcoin アドレスと異なる存在が、コントラクトアドレスである。Ethereum ではスマートコントラクトと呼ばれるものが作成できる。これは人が持つものではなく、コードとして定義された関数である。Ethereum はそれそのものがブロックチェーンでありながら、Ethereum ネットワーク上でコードによって支配された世界を築く土台であろうとしている。そしてコードによって支配された世界に当たるものがこのスマートコントラクトである。また、このスマートコントラクトにつけられた Ethereum ネットワーク上でのアドレスがコントラクトアドレスとなる。スマートコントラクトはよりフレキシブルな非中央集権の世界を作ることができる。Bitcoin は金にあたるトークンのやりとりのみが行われているのみだった。その一方で、スマートコントラクトは例えば管理会社のいないギャンブル市場を作ることが出来るのだ。ここでは Augur という Ethereum 上で動いているスマートコントラクトを例に述べる。Augur の参加者はまず、賭けに関するトピックを生成し、それをブロードキャストする。それに興味を持った参加者が賭けを行うため、作成されたトピックに存在する選択肢から一つを選び、賭けたい量の Augur トークンを賭ける。もしこれが競馬のレースであったとするならば、レース終了後にこの事実を認定するフェーズに入る。事実はレース終了後に選択肢の中から正しい選択肢に対して賭けることで行われ、レース終了後でもっとも多く賭けられた選択肢が事実として認定される。その後、争議ラウンドが設けられ、これに対する異議申し立てを行える期間がある。そして最終的な決着を見た結果に基づいて払戻金が支払われるという仕組みになっている。この際、トピックを生成することに関してインセンティブとなるように生成者には一定の Augur トークンが支払われる仕組みを持っている。さらに、レース後の投票においても最終的な結論に至らない場合、チェーンがフォークすることも前提にした仕組みを持っている。そしてこの賭けのサービスを応用すると、保険機構も作れるのだ。「A さんが怪我を負うか」というトピックにに A さんがずっと「怪我を負う」賭け続け、A さん以外が「怪我を負わない」に賭け続けるとする。すると A さんが怪我を負った時、賭けに勝ったお金として保険金に当たる今まで「怪我を負う」に賭け続けた失ったトークンが戻ってくるのだ。この場合、怪我の認定について誰が行うかなどの曖昧な点が残されているが、原理としては行うことが出来る。このように、スマートコントラクトは単なる送金よりもフレキシブルなコードによって支配された非中央集権の世界を作ることの可能な技術なのだ。

3.5.3 solidity とチューリング完全

solidity は前述のスマートコントラクトを記述する言語である。そして、フレキシブルな世界実現のため、solidity はチューリング完全な言語となっている。例えば、貨幣のような価値を持つトークンの存在を前提としたスマートコントラクトには ERC20 という基準が存在するが、これについての関数とイベント宣言の記法について見てみる。これは、この ERC20 に沿っていれば、Ethereum ネットワーク上でトークンとしての役割を果たせると言えるものである。

ソースコード 3.4: ERC20 を満たすために必要な関数とイベントの宣言文

```
function totalSupply() constant returns (uint256 totalSupply);
function balanceOf(address _owner) constant returns (uint256 balance);
```

```
function transfer(address _to, uint256 _value) returns (bool success);
function transferFrom(address _from, address _to, uint256 _value) returns (bool
    success);
function approve(address _spender, uint256 _value) returns (bool success);
function allowance(address _owner, address _spender) constant returns (uint256
    remaining);
event Transfer(address indexed _from, address indexed _to, uint256 _value);
event Approval(address indexed _owner, address indexed _spender, uint256 _value);
```

function totalSupply:当該トークンの供給量を取得する。

function balanceOf:指定した_owner アドレスの残高を取得する。

function transfer:呼び出し主のアドレスが所有するトークンから_value の量を_to のアドレスへ送金する。

function transferFrom:トークンを_from のアドレスから_to のアドレスへ_value の量を送る。このコントラクトの呼び出し主はトークンの管理者であり、_from アドレスは approve によって許可された範囲内での送金となる。

function approve:管理者のトークンのうち、_value までの値の分だけ_spender のアカウントから使うことを許されるように管理者が宣言する。

function allowance:指定した_owner は_spender アドレスに対してどれだけの量のトークンを支払うことを許可されているかを取得する。

event Transfer:トークンが送られた段階で発火し、_from から_to へ_value のトークンが移動したことを出力する。

event Approval:approve が呼ばれた際に発火し、承認情報を出力する。具体的には、_owner から_spender へ_value の量のトークンを流すことを承認したことを出力する。

以上が solidity の書き方である。また、solidity の関数にはセッター関数とゲッター関数の 2 種類が存在する。セッター関数は提出したトランザクションがブロックに入った時に実行されるもので、gas が必要となる。一方、ゲッター関数はその関数の実行コマンドを押した瞬間に実行され、結果が返ってくる。この処理にガスは必要ない。ブロックチェーン上の状態を変更するかしないかでこの 2 種類が存在し、ERC20 では transfer などが前者、totalSupply などが後者に分類される。また、Transfer や Approval といったイベントは関数が実行されている時に event を送出するコードによって発火し、クライアントによってこれを監視することが出来る。そして書かれた solidity は solc と呼ばれるコンパイラによって EVM(Etherum Virtual Machine) が解釈可能なバイトコードへと変換され、スマートコントラクトとしてブロックチェーンネットワーク上へデプロイされる。以上がチューリング完全な言語である solidity の説明である。

3.5.4 μ Raiden

μ Raiden は Ethereum におけるオフチェーン技術の一つである。ビットコインのオフチェーンとは違い、これはスマートコントラクトによってオフチェーンを管理する。最初にトランザクションとしてのデポジットをスマートコントラクト上へ提出する。その後、ブロックチェーンとは無関係の通信、http(s) 通信などによって最終的にこのコントラクトから払い戻されるトークン量を決める。例えば A が 100 トークンをデポジットしており、これを 10 トークンずつ B へ送信するとしよう。この時、最初に A は 100 トークンをデポ

ジットする必要がある。これにより、直接オフチェーンにてトークンが送信されるペイメントチャンネルが開かれる。その後、10 トークンを利用したことを証明する署名値を B へ送ることでトークンを送金したことが証明される。また、μ Raiden には B から A へ逆方向にトークンを送ることは出来ず、一度送ったトークンバランスからトークンの受取り手の取り分を減らすことは不可能である。最後に、トークンバランスに基づいてスマートコントラクト内の関数が呼び出され、トークンの分配が行われる。
この分配は二つの方法があり、トークンの送り主による署名値無しのトランザクション発行によるものか、トークンの受取り手による署名値ありのトランザクション発行によるものの二つである。最初に署名値ありのトランザクションによる分配について述べる。トークンの受取り手はがデポジットされたトークンの内から自身へ支払われたトークンをすぐにオンチェーンで使えるものにしたいという動機などから、即座に取引を終了することが可能である。この取引終了はトランザクションとしてブロックチェーン上に送信され、ブロックに含まれた際に処理される。仮にトランザクションプールが肥大化していて、トランザクションが詰まっていてこのトランザクションが長い間処理されない場合、次に記述する送り主による取引終了のフェーズに移る可能性がある。次に、署名値なしのトランザクションによる分配について述べる。一預けたデポジットのうち転送していないトークンを取り出したいという動機などから、買い手はチャレンジピリオドを経たのちに取引を終了することが可能である。一般に、一方的にオフチェーンでのトークンの転送が行われるため、送り手は受取り手から署名を得ることをしない。従って、この取引終了のリクエストは相手側の署名なしに発行することのできるトランザクションによって行うことが可能である。しかしこの時、送り手がオンチェーンへと取引終了のトランザクションを提出する場合、送り手が全くトークンを支払いを行なっていない状態の署名を提出することが可能である。そこで仮に送り手が不正なトランザクションを提出した場合は、受取り手が先に現在のトークンバランスを用いた取引終了のトランザクションを提出できるようにするべきである。そしてこの猶予の期間がチャレンジピリオドと呼ばれる期間である。このチャレンジピリオド内に受取り手が反応しなかった場合は、さらに後に発行される送り手のコントラクトによって、送り手の主張するトークンバランスで最初のデポジットが分配される。このように、チャンネルのクローズ時のトークン分配は行われる。また、これらの流れを一つの図にしたものか下図 3.1 である。μ Raiden はスマートコントラクトを使用しているので、よりフレキシブルにオフチェーン取引中の中間でトークンを引き出すことができる。また、オンチェーンのトランザクションによってオフチェーンにデポジットしているトークン量を増やすことも可能である。そして ERC20 や ERC223 に準拠した Ethereum ネットワーク上で動くトークンに対して全てで動くように設計されている。

3.6 まとめ

この章はブロックチェーン技術全般についての仕組みやその問題点、Bitcoin や Ethereum の特徴などについて述べた。途中ではトランザクションの処理の上限を緩和するための技術であるオフチェーン技術についても触れた。また、Ethereum は最初にブロックチェーンが実装された Bitcoin よりもフレキシブルな非中央集権のコードベースで動く世界を作られる可能性を秘めていることを述べた。

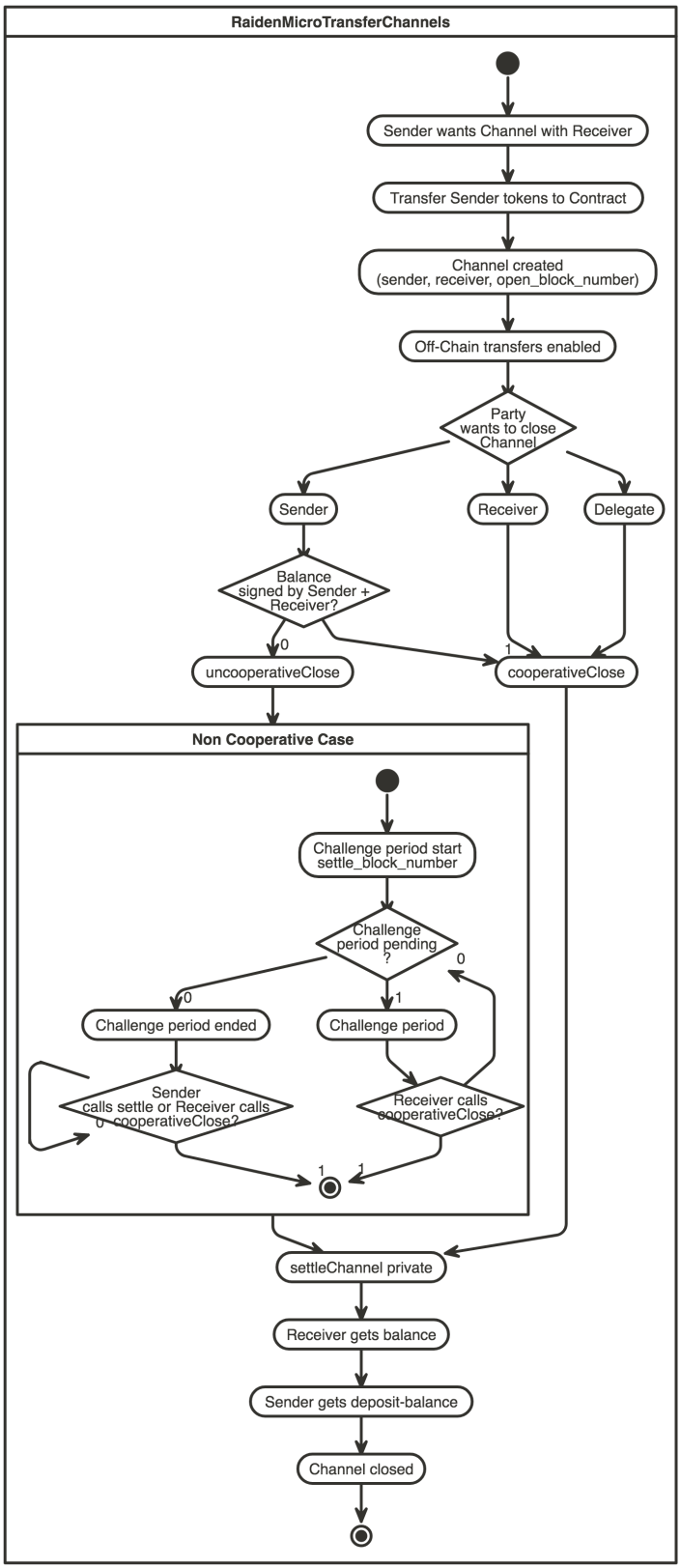

図 3.1: μ Raiden のプロセス

第 4 章

MIWWG:支配者の存在しない IoT データ市場

本章では、本研究で提案する IoT データ市場である MIWWG についての要件と、そのデータ市場における取引のプロセスについて記述する。また、MIWWG とは ”MIWWG works without governor”を表し、再帰的頭字語を使っている。

4.1　MIWWG が達成する目的

本研究の最終的な目標はオープンな IoT データ市場の参加者が、参加者自らの意思によってルールや管理団体を設立できるものである。詳細な理由については 4.2.1 項にて後述するが、今のクローズドな IoT データ市場の方式を拡張することでこの理想の実現を目指すことは難しい。一方で、オープンな IoT データ市場の方式を拡張することでこの理想の現実を目指すことは可能であると考える。この時、もっとも最終的な理想の市場を実現するため、もっともオープンな市場を実現することが本研究の MIWWG が達成する目的である。今後この P2P 通信によって構築された MIWWG を基にした、ルールや管理団体の設立の方法について議論や実装が行われることを期待するものである。

4.2　市場の要件

本節では、前節に記述した達成する目的のため、より具体的に市場が満たすべき要件を述べる。前半の 1 2 項に関しては機能要件であり、後半の 3 6 項に関しては非機能要件である。前半は MIWWG の基本理念を継承したものであるか否かを測るものとして、後半は MIWWG の評価指標として MIWWG が改良されて行くこの先も使用されうるものであることを意識して記述した。

4.2.1　中央集権装置を必要としない通信モデルによる市場の創造

本市場において、中央集権組織である管理者の存在はオープンな市場である上でもっとも不要なものである。管理者は市場を一存で動かせるため、当然ながら巨大な力を持ち得る。そしてこの大きな力は市場の前提条件を簡単にひっくり返せたり、市場の取引に対する疑義を生じさせる原因となる。全員が透明に見ることが可能なルールによる支配を行えることにより、この危険性への懸念や疑義を払拭することが出来る。この中央

集権組織支配の排除を目指すとき、既存の代表的なサービス提供方式であるサーバ・クライアント型ではサーバ上でサービスが提供される以上、このサーバの管理者が市場に管理者となってしまい目標の達成が難しい。そこで例えば P2P 通信により、市場の参加者の全員が対等な立場で市場を形成することを考える。P2P 通信は特定の管理者を不必要とするが、この P2P 通信上で同意を得た上で管理者にあたるノードを選ぶことは可能である上、P2P 通信上でルールやプロトコルを作ることは可能である。ここで規制が丁度良いルールや管理団体の存在する市場を目指すとき、もっともクライアント・サーバ型を使ったクローズドな市場から目指すことは難しいが、逆に P2P 通信型を使ったオープンな市場から目指すことは可能であるという現状が見えてくる。そして MIWWG は丁度良い規制の市場を目指す最初の段階として、もっともオープンでルールも管理団体も存在しない IoT データ市場を P2P 通信で実現することを目的としている。よって MIWWG の機能要件の一つとして、P2P 通信を始めとした中央集権装置を必要としない通信モデルによる市場の創造が挙げられる。

4.2.2 データの売買

本市場において、データの売買が可能である必要がある。ここで売買とは何かを日本国の民法 555 条から引用して考える。「売買は、当事者の一方がある財産権を相手方に移転することを約し、相手方がこれに対してその代金を支払うことを約することによって、その効力を生ずる。」つまり、データ売買に関する取引が成立したとき、データの売り手が買い手へデータを提供する必要がある。また同時に、買い手は売り手へ代金を支払う必要がある。そして、これらには契約と同時に上記二つの必要な要件を履行する義務が生じる。但し、これらは法律的な売買の定義とその解釈であり、実際にデータの売買契約には以下のプロセスが必要となる。また、今後そのプロセスを遂行するために作られたモジュールを呼ぶための名称を付記する。

1. 売り手がどのようなデータを売りたいか、またその値段などを示し、データを出品する。データ出品モジュール
2. 買い手が出品されたデータを閲覧する。：出品データ一覧取得モジュール
3. 買い手が閲覧したデータの中から欲しいデータを選び、売り手へ売買契約を申し込む。：売買申し込みモジュール
4. 売り手は申し込まれた契約に対し、応じるか否かの返答をする。：売買契約前確認モジュール
5. 売り手が申し込みに応じた場合、売買契約を成立させる。：売買契約作成モジュール
6. 買い手が売り手へ代金を支払う義務を遂行させる。：支払い義務遂行モジュール
7. 売り手が買い手へデータを転送する義務を遂行させる。：データ転送義務遂行モジュール

以上が必要なプロセスであり、データの売買に必要な要件である。

4.2.3 取引に関する秘匿性

ここでは 2 点について述べる。A. データに関して、売り手が意図しない相手へのデータの公開は行わない。B. 取引の内容について、トークンやデータの取引上必要な情報以外は極力ブロックチェーン上に載せず、公開しない。まず A についてだが、データは情報商品であり、その情報は一般に入手できないため、価値が存在するものである。そのデータが本市場の上から意図しないところへ大量に漏れるとすれば、それは本市場がデータの価値を毀損していることにつながり、本市場が意味の少ないものになってしまう。よってブロックチェー

ン上へデータを送るなど、マイナーを含む買い手以外の人間に対して売り手が意図しないところでデータが公開されることは避けなければならない。次に B についてだが、例えばデータを取得するためのアクセス先かアクセス元を公開しなければならないとする。例えばデータを売り手が買い手へプッシュするのか、買い手が売り手にプルするのかについて議論をしてみる。この時 http(s) 通信を使い、 curl などでデータをプッシュかプルをする際は必ず売り手か買い手が IP アドレスをブロックチェーン上へ暴露しなければならない。そうでなければ、ノードが見つからないためだ。ここでは平均して 1 つの事業者が平均で m 個のデータノードを所有しており、1 つのデータノードは平均して n 個の事業者にデータを売ると考えられる市場を仮定する。この時、データ受け取り手の IP アドレスを暴露する場合は 1 人の事業者に対して $m \times n$ 個の IP アドレスを暴露する必要がある。一方でデータの送り手の IP アドレスを暴露する場合は 1 人の事業者に対して $\frac{1}{m \times n}$ 個のアドレスを暴露する必要がある。するとこの場合、$m \times n \geq 1$ であればデータの送り手の IP アドレスを暴露する方が、事業者に紐付く情報が少なく済む可能性がある。勿論、これに関しては作成する市場の目的によって異なる。また、IP アドレスよりも一時的に使用するために生成されたメールアドレスなどの方が、より秘匿性は高い場合も存在するであろう。ここでは、取引に関してなるべく他者から秘匿されるべき項目についてあげ、記述を行なった。

4.2.4 売買方法の決定可能性

オープンでルールや支配組織の存在しない市場は、売り手が広場に農作物を持ち寄って勝手に出来上がる市場と同じようなものである。そこにおいては当然、売り手は好きな方法で好きな値段をつけて商売ができる。これと同じように、本市場の売買方法はフレキシブルである方が良い。例えば 4.2.2 項で記した「支払い義務遂行モジュール」と「データ転送義務遂行モジュール」の二つについて考えてみる。民法による売買の定義を参照すると、本来は売買契約と同時に、売主の財産権移転義務と買主の代金支払義務は同時履行される。しかし、データとトークンを同時に一瞬のラグもなく送信し合うことは技術的に難しい。そこでデータとトークンのどちらを先に送信するかについて議論が起こる。売り手のデータの到着を確認して、買い手のトークンを送るのか、あるいはその逆か。後払いを選んだ場合、最初に売り手がデータを投げ、それに対して買い手は支払いを行う。その後はトークンの移動を売り手が確認してから新しくデータを投げることとなる。一方で前払いを選んだ場合、最初に買い手がトークンを投げ、それに対して売り手はデータを送る。その後は買い手は送られてきたデータを確認してから新しいトークンを投げ、必要なデータを取得することとなる。前払いが良いか後払いが良いかは、売るべきデータの種類に依存すると考えられる。例えば 3 秒に 1 回送信されるデータであり、データが何個も連続して時系列に送られることによって初めて価値のあるデータというものが存在するだろう。例えば高速道路を定点観察している写真がこのデータの内容である場合だ。この時は、データに対して後払いを認めた方が得であると考えられる。何故なら、ユーザはそのデータの内容を見てから、本当に欲しかったデータ形式やデータ内容であるかを精査できるからである。一方。例えばある地点の気温と湿度のデータを 10 分に 1 回送信しているデータノードが存在するとしよう。この時に後払いを認めてしまうと、一度しか情報が入らない人は常に一度のみのデータを貰い、それに対するデータを払わずに取引を中断してしまう可能性がある。このように、前払いか後払いかについてでも、データの内容によって売り手の希望は変わってくる。同じように、何個のデータに対してどれだけのトークンを支払い、支払いのラグはどこまで認めるのかについて売り手がよりフレキシブルに決められることが要件の一つである。

4.2.5　大量な IoT データのリアルタイム処理

本研究において、IoT データとは細かい間隔 (1 10 秒) でリアルタイム流れてくるデータのことを呼ぶ。蓄積データが対となる概念であり、ある都市の過去 1 年間の気温データなどはリアルタイムではなく既に存在しているデータであるので、蓄積データと呼ぶ。これに対し、現在のある都市の気温はリアルタイムで取得できるデータであり、IoT データと呼ぶ。この IoT データの売買を裁くには、大量の処理を裁く必要が出てくる。少なくとも、PoW(Proof of Work) を基にしたブロックチェーンのオンチェーン技術のみでは裁くことができない。その場合はオフチェーンを利用することが考えられる。また、将来的には一部をオンチェーンで PoA(Proof of Authority) を利用し、一部の管理団体がトランザクションを速く裁くことも必要になるかもしれない。PoA とはトランザクションのマイニングを一部の有資格者が行える仕組みのことで、一定のハッシュパワーでトランザクションの承認が行われる方式である。この方式は、承認者の数が多いほど捌けるトランザクションの数が増えるという利点を持っている。一方で、中央集権型の統治へ逆戻りしてしまっている様子も伺える。オンチェーンやオフチェーン· それに関わる政治的な要素は様々あるが、IoT データというリアルタイムデータを扱う以上、大量データのリアルタイム処理を行えることは要件の一つである。

4.2.6　ダウンタイムの非存在

IoT データ市場は IoT の発展に伴い、とても大きな市場規模を持つようになる。データ市場に限らない数値であるが、2022 年の IoT 市場規模は 11.7 兆円と言われている。(http://www.itmedia.co.jp/enterprise/articles/1809/13/news
このようなプラットフォーム上で大きな価値のあるアプリケーションが動いている場合は、ダウンタイムが起きると多額の損害が生まれてしまう。その状況の中で、2 章で述べたように AWS の SLA は 99.99% であり、年間では 52.56 分落ちている可能性があるのだ。そのわずか一時間弱の間であっても、IoT データ市場の規模の大きさを考えるとダウンタイムは看過できない。また機械的のハードウェア的な故障のみならず、人為的なバグや人為的なオペレーションミスも一つのサーバ上でサービスが動いている場合は考えうる。そこで P2P 通信を始めとした方法で、ある事業者のサーバが壊れてもネットワーク全体には影響を及ぼしにくいサービスモデルであることが望ましい。P2P 通信であることは要件にはならないが、ダウンタイムの存在がほぼ否定されるほど、この市場が動くネットワークがダウンタイムを起こしにくくすることは要件の一つである。

4.3　取引のプロセス

本節では、MIWWG における IoT データ取引のプロセスを示す。

4.3.1　データ陳列

最初に、データの売り手が自分の持っているデータの種類を MIWWG ネットワーク上へ公開する。この際、取引に必要な事項である値段やデータが送信される間隔といった値を同時に提出する。しかしデータについての詳細な説明書きや例を大量の文字によって記述する場合があった場合、マイナーへ大きな量の手数料を払う必要が生じる。その際はデータに関する詳細を記述した URL を記述し、その URL に詳細なデータに関する記述を行うという方法を取ることも可能である。売り手は自身の持つデータをブロックチェーン上へと送信し、これがブロックに含まれた瞬間からデータが全ノードへと知れ渡ることとなる。

4.3.2　取引開始

次に、買い手が陳列されたデータを閲覧し、必要なデータについて取引を開始する。この時、一般的なオフチェーン技術の特徴から、買い手は今後どの程度の期間でそのデータが必要であるかについて考慮し、その期間の間に使われる分のトークン量をデポジットする必要がある。このデポジットについては、相手の同意のない限りデータの買い手側が即座に引き出すことはできない。取引開始のトランザクションをブロックチェーン上へ送信し、それがブロックに含まれた瞬間が取引開始の瞬間である。トランザクションがブロックに含まれた後、http(s) 通信などによってデータを読み込む際の ID とパスワードをデータの販売元へ送信する必要がある。

4.3.3　データ販売とトークン転送

本項を含む、オフチェーンの文脈においてトークンの転送とは、買い手がそのトークンバランスに同意したことを示す署名値を売り手へ送信することである。取引開始後、トークンが買い手から売り手へとデータの送信間隔と同じ間隔で送信される。データ転送を売り手が止めた場合、買い手はトークンの転送を止める。逆に、トークンの転送を買い手が止めた場合は、売り手はデータ転送を止める。この際、故意に転送を止めた場合でなくとも、ネットワークの一時的な不通によってデータやトークンの転送が行われなかった場合も同様に相手側は取引を停止する。従って、データ陳列の際にどの程度の支払いの遅れまでを売り手が許容するかという内容や、売り手のネットワーク状況によってどの程度のデータ転送の遅れまでを買い手が許容するかという内容についてをデータ陳列の際に記述しておくべきである。このデータ転送は http(s) 通信によって行われ、ブロックチェーンネットワークとは全く関係のない動きをする。従って、これらのデータ販売や転送はブロックチェーンの処理能力に囚われない。

4.3.4　取引終了

買い手がトークンの転送をやめた場合や売り手がデータの転送をやめた場合は取引の途中であっても取引が終了する場合がある。逆に、デポジットの全ての転送が終わった場合はデータの売り手によって取引は終了される。いずれでも、買い手か売り手が取引終了に関するトランザクションがブロックチェーンネットワークに送信することで、取引は終了を迎える。

4.4　まとめ

本章に記述したプロセスを一つの図で示したものが以下の図**??**である。買い手と売り手がどちらも取引終了を主張した場合のプロセスを示している。

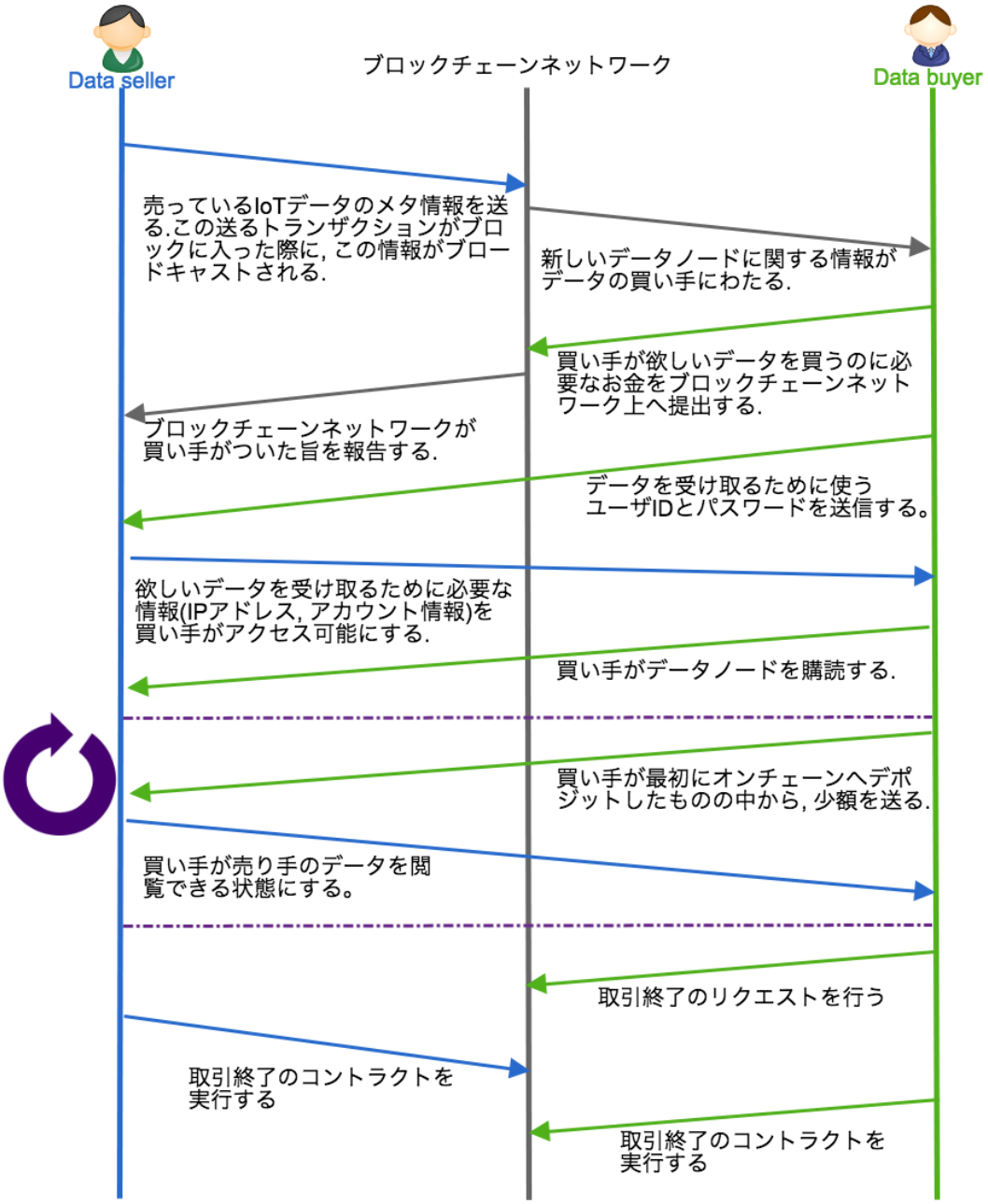

図 **4.1:** MIWWG の取引プロセス

第 5 章

設計と実装

本章では, 前章にて述べた MIWWG を実現する際の設計と実装について述べる。また、本研究においては大きく分けて 3 つの技術を用いている。市場における貨幣の役割を担うブロックチェーン技術、そのブロックチェーン技術の弱点であるトランザクション流通量の上限を引き上げるオフチェーン技術、IoT データを実際に送信する役割のメッセージングシステム技術の 3 つである。これら 3 つのうち各々をどのように用いたかについて、独立した節にて説明を行う。

5.1 設計

最初に、本節では設計について述べる。

5.1.1 システム構成

以下に、システム設計図 5.1 を示す。

5.1.2 ブロックチェーン技術

本研究において、ブロックチェーンのトークンが貨幣の役割を担い、ブロックチェーンはその元帳の役割を持つ。一般に、ブロックチェーンはコントラクトを生成することが出来る。そのコントラクト上にて、市場に存在するデータの管理やトークンの移動を行う。データに関する情報のうち、多くの人にとって重要な定量的情報を列挙し、それをコントラクトの中に記述する。本研究では以下の項目を定量的情報として入力させることとした。

- データの名前
- 1 つのデータあたりの値段
- データが送られてくる間隔
- トークン送信の遅延に関する許容時間
- 前払いで必要なトークンの量
- データの売り手のアドレス
- データに関するその他の説明がある場合、それを記述する URL

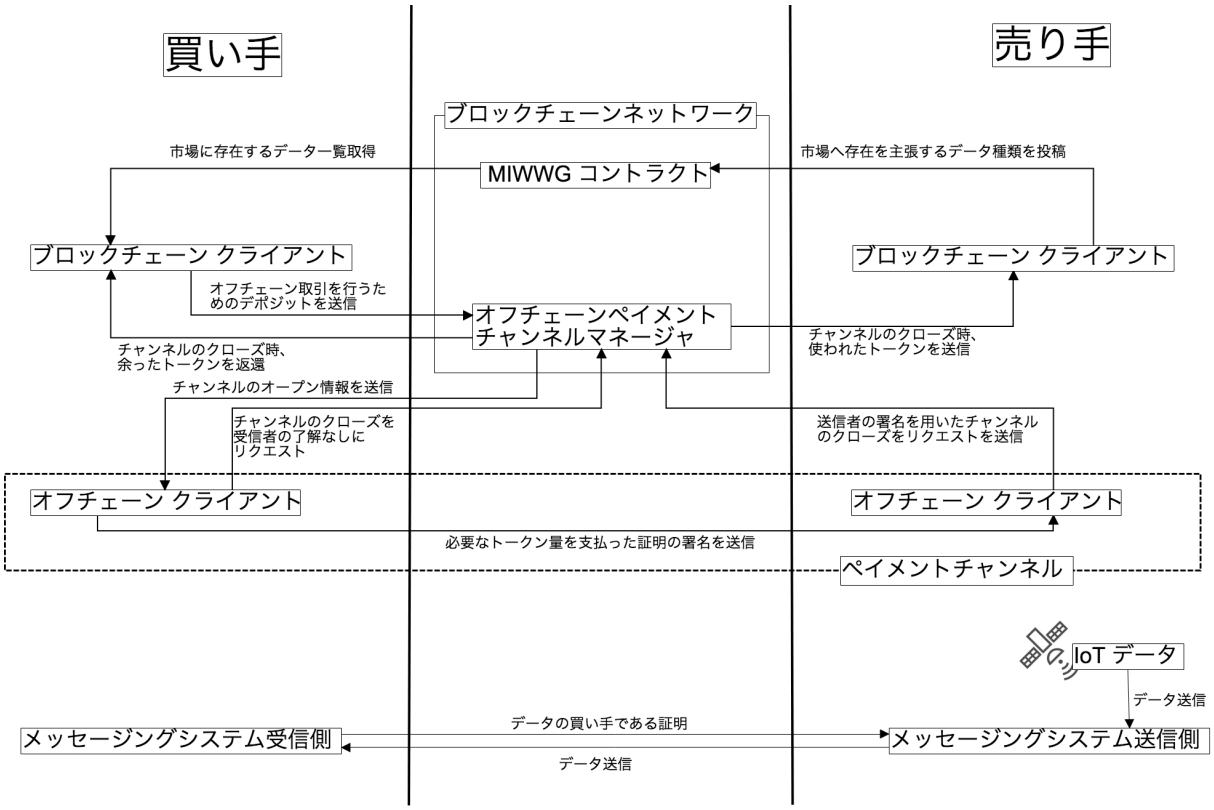

図 5.1: システム設計図

1 つのデータあたりの値段とデータが送られてくる間隔、データの売り手のアドレスはデータの買い手がトークンを利用する際に必要な情報である。また、買い手が環境の悪いネットワーク環境下からデータを買おうとする場合、トークン送信の遅延に関する許容時間は重要なファクターとなる。買い手に契約を終了する気持ちがなくとも、売り手がトークン送信の見送信を検知し、契約が終了となる可能性があるためである。その場合、もう一度契約をオンチェーンの処理によって開始する必要があり、これは本来不必要であったマイナーへの手数料を必要とする。前払いで必要なトークンの量は、契約を開始した後に直ぐに買い手がトークンを未送信にしてしまい、買い手がデータを遅延の許容時間分だけ無料で受信するのみで契約を終了してしまう行動を抑制するためのものである。また、これらの定量的情報以外にデータに関する詳細な説明を記述する場合は、そのデータに関する情報を記載した URL をトランザクション内に記述する。このことにより、このデータの存在を市場へ主張する際のトランザクションサイズを小さくする狙いがある。一方、同じ URL であってもそこに記載する内容はブロックに入った後も変更可能である。従って、取引における重要な事項をこの URL に記述したとしても、それは信頼できるものではない。あくまで URL の記述内容は他の項目にプラスして、データに関する内容を理解するために使われるものとなる。

5.1.3 オフチェーン技術

本研究において、オフチェーン技術はブロックチェーンの処理能力を補完するために使われる。IoT データは短い時間間隔で流れてくるデータであり、ブロックチェーンの処理能力では追いつかないためだ。データ

取引の開始時にオンチェーンにてデポジットを行い、ペイメントチャンネルをオープンし、その情報がオフチェーンクライアントへ渡る。そしてデータの買い手は売り手へ新しいトークンバランスによる署名値を提出することで、データの売り手へトークンを渡す。最後に、データの買い手か売り手のいずれかがチャンネルをクローズし、最終的にデポジットされたトークンを分配する。

5.1.4 メッセージングシステム

本研究において、メッセージングシステムは IoT データの送信に使われる。一般に、メッセージキューシステムに基づく分散メッセージングシステムは次のような特徴を持つ。

- 非同期のメッセージ送信が可能である。これにより送信側と受信側とのサービスが疎結合になり、片方の処理の遅延がもう一方に影響を与えるリスクが減る。
- 送信を行う一つのノードが落ちたとしても、他のノードが直ぐに落ちたノードの代わりの役目を担う。これにより、可用性が高いことが担保される。
- 送信側で新しくノードを追加した時、新規ノードを含めた協調動作が始まる。これにより、拡張性が高いことが担保され、より多くのメッセージ処理を裁くことが出来るような拡張を簡単にすることが出来る。

これらの特徴から、今回の IoT データ市場の構築にはメッセージングシステムを使うことは最適である。送信者と受信者のシステムが疎結合であり、各々の責任によってデータの送受信が可能なためだ。また、メッセージングシステムによっては CAP の定理においてどの要素を重視するかをパラメタで指定することも可能であり、データの送受信者にとってよりフレキシブルな設定が可能となっている。
また、メッセージングシステムは kerberos や簡単なユーザ ID/パスワード認証などを用い、データの買い手のみへデータの提供をしなくてはならない。そしてトークンの取引が停止されたことが検知された時からなるべく早く、データの送信を不許可にしなくてはならない。

5.1.5 実装

5.1.6 システム構成

以下に、システム構成図 5.2 を示す。

5.1.7 ブロックチェーン技術

本研究では、ブロックチェーン技術の中から Ethereum を利用する。また、その Ethereum クライアントとして、go 言語による実装である go-ethereum(以下、geth) を利用した。ここで geth のオリジナルコードとは、geth の github レポジトリ (https://github.com/ethereum/go-ethereum) の Constant (v1.8.20) のことである。solidity は 0.4.17 のバージョンを利用している。最初に開発したトークンの中身について記述する。採用したトークン規格は ERC20 に次ぐトークンの規格である ERC223 を考慮しつつ、一部を改変して利用した。これは ERC223 が本研究で採用するオフチェーン技術である μ Raiden がより薦める方式であるためである。既存の ERC20 と比べて、コントラクトアドレスに対して誤ってトークンを送った時に取り戻せる

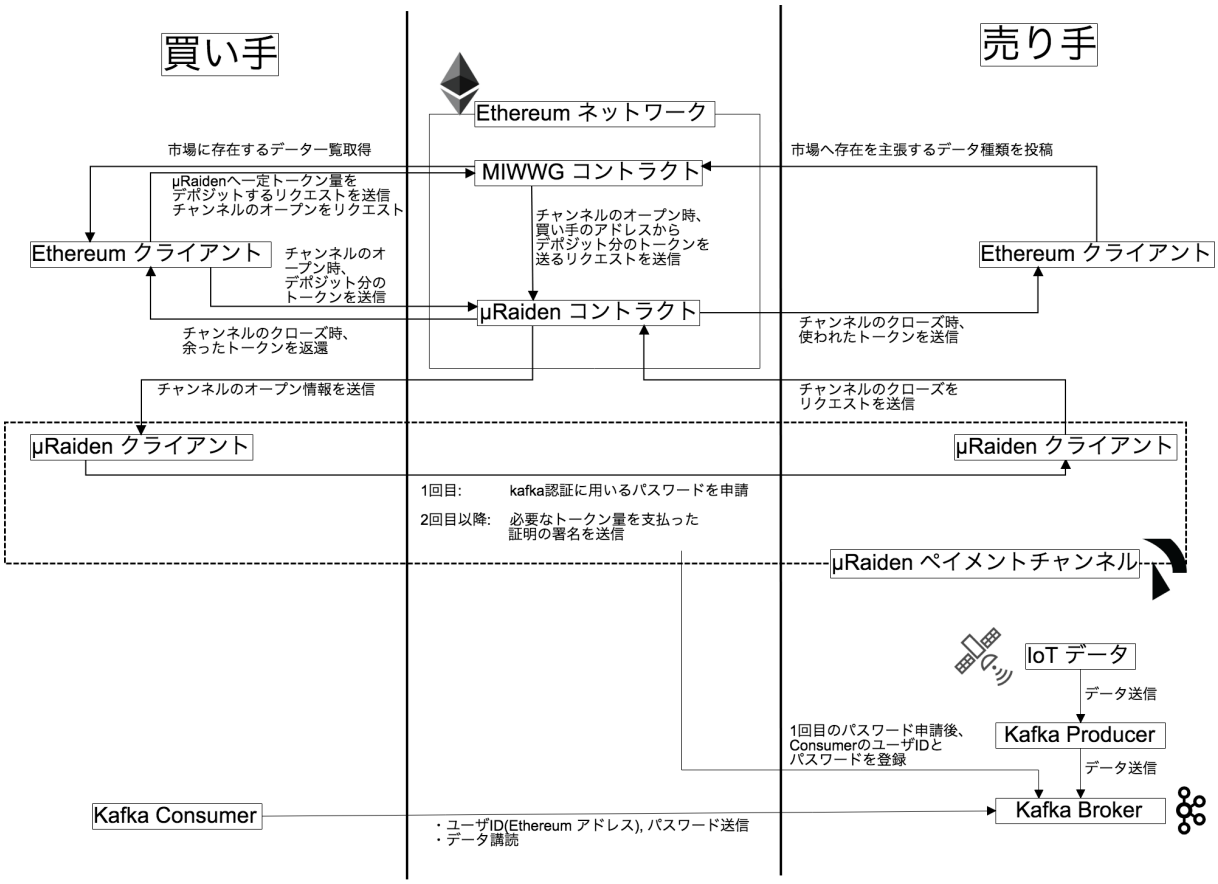

図 5.2: システム構成図

tokenFallback 関数が追加され、コントラクトに対するトランザクションとユーザに対するトランザクションを同じように捌けるようになっている。これにより Ethereum の掲げる理想である相手がコードか人間か気にせずにトークンのやり取りが出来るという世界をより体現しているものである。この ERC223 の一般的な実装物は github のレポジトリである Dexaran/ERC223-token-standard(https://github.com/Dexaran/ERC223-token-standard) に存在する。この元のソースコードに加える形で本研究では以下の構造体によって IoT データを格納可能とした。

ソースコード 5.1: solidity によるデータを保持する構造体

```
struct DataNode {
  string name;  // データの名前
  uint256 price;  // 1つのデータあたりの値段
  uint256 interval;  // データが送られてくる間隔
  uint256 delay_permission_time;  // トークン送信の遅延に関する許容時間
  uint256 prepaid_value;  // 前払いで必要なトークンの量
  address author;  // データの売り手のアドレス
  string explanation_url;  // データに関するその他の説明がある場合、それを記述する URL
  uint256 data_index;  // 本データがデータ市場において一意に定まる ID。1から順番にインクリメントされていく。
```

```
}
```

以上が IoT データの種類を格納する構造である。また、この他に Ethereum トークンを本トークンへ変換することのできる機能を実装した。本トークンは Ethereum の 1wei = 1MI(本トークンの通貨単位) として Ethereum ネットワーク上で動くものである。さらに、一般的な transfer(_to, _value, _data) の関数をラップする buyData 関数を作ることによって、μ Raiden クライアントから何のデータを買いたいかについて、買い手からの情報を得る実装を行なっている。以下がその様子である。

ソースコード 5.2: geth の EIP155 のフォーク確認機能の削除 params/config.go

```
uint256[] user_bought_log; // ユーザの購入履歴を格納
function buyData(
    address _to,
    uint256 _value,
    bytes _data,
    uint256 data_index)
    public
    returns (bool)
{
    user_bought_log.push(data_index); // ユーザの購入履歴に追加
    emit boughtData(msg.sender, data_index);  // ユーザの購入が行われたことを示す
        eventを送出
    bool ret = transfer(_to, _value, _data);  // ERC223対応のtransfer関数を実行
    return ret
    }
```

これにより、一つのトランザクションによってデータの購買とそれに関するペイメントチャンネルを開くという二つの行為ができるようになっている。

次に今回使用した geth に加えた変更について述べる。今回はプライベートネットワークで動作を検証するにあたり、geth がμ Raiden の発行する EIP155 を考慮した署名のトランザクションを捌かないという事象が発生した。これはリプレイアタックを防ぐためのものであり、Ethereum のブロックナンバーが 2675000 以降のものについてのみ EIP155 を適用する実装がなされていた。しかし、これではトランザクションが裁かれ始めるまでに大量の時間を要してしまうため、この機能を簡単に削除した。

ソースコード 5.3: geth の EIP155 のフォーク確認機能の削除 params/config.go

```
227 // IsEIP155 returns whether num is either equal to the EIP155 fork block or
    greater.
228 func (c *ChainConfig) IsEIP155(num *big.Int) bool {
229     //return isForked(c.EIP155Block, num) // 既存の判別箇所を削除
230     return true // 全てのEIP155署名のトランザクションが裁かれるように変更
231 }
```

5.1.8 オフチェーン技術

オフチェーン技術について、本研究ではμ Raiden(https://github.com/raiden-network/microraiden) の 0.2.0 を使用した。このμ Raiden について、いくつかの変更を加えている。最初に、ペイメントチャンネル

をオープンするために MIWWG コントラクトへ transfer 関数を実行する際、独自の関数である buyData 関数を呼び出している。

ソースコード **5.4:** μ Raiden クライアントの変更 client/client.py

```
def open_channel(self, receiver_address: str, deposit: int):
    data = decode_hex(self.context.address) + decode_hex(receiver_address)
    tx = create_signed_contract_transaction(
        self.context.private_key,
        self.context.token,
        # 'transfer', //オリジナル部分
        'buyData', // 追加部分
        [
            self.context.channel_manager.address,
            deposit,
            data,
            data_index // 追加部分
        ]
    )
```

上記と同じ変更をペイメントチャンネルへのデポジットを増やす際に使用する topup 関数にも加えている。また、デフォルトの設定で用いると ethereum のテストネットワークへと繋がってしまい、プライベートネットワークに繋がらない。よって以下の変更を加えた。

ソースコード **5.5:** ネットワーク設定の変更 config.py

```
 40 # network-specific configuration
 41 NETWORK_CONFIG_DEFAULTS = {
 ...
 63     # private MIWWG //63行目から67行目まで追加部分
 64     98: NetworkConfig(
 65         channel_manager_address='0xd0c19d7d41ae4a5561d2289d997da00b9de1bf73',
 66         start_sync_block=0
 67     ),
 ...
104     #NETWORK_CFG.set_defaults(3)  // デフォルトのネットワークID
104     NETWORK_CFG.set_defaults(98)  // 今回使用するネットワークID
```

また、μ Raiden はバックエンドの DB として sqlite を使用している。この DB を使って、のちにメッセージングシステムにてユーザ ID とパスワードを使用した認証を行う。そのパスワードを保持する目的と、μ Raiden では想定されていない同じ送り手と受取り手のアドレスに対して複数の別のペイメントを使用することから、DB 構造について変更を加えている。

ソースコード **5.6:** DB 構造の変更 channel_manager/state.py

```
CREATE TABLE `channels` (
    `sender`              CHAR(42)          NOT NULL,
    `data\_index`             INTEGER           NOT NULL,  # 追加
    `password`            CHAR(42)          NOT NULL,  # 追加
    `open_block_number` INTEGER           NOT NULL,
```

```
    `deposit`               DECIMAL(78,0)    NOT NULL,
    `balance`               DECIMAL(78,0)    NOT NULL,
    `last_signature`        CHAR(132),
    `settle_timeout`        INTEGER          NOT NULL,
    `mtime`                 INTEGER          NOT NULL,
    `ctime`                 INTEGER          NOT NULL,
    `state`                 INTEGER          NOT NULL,
    `confirmed`             BOOL             NOT NULL,
    PRIMARY KEY (`sender`, `open_block_number`, `data_index`)  # 最後の要素について、追加
);
```

またこれに伴い、この DB を利用するコードについても全て変更を加えている。他の機能として、パスワードを登録する際にアクセスする api のエンドポイントを作成している。Web サーバは gevent の pywsgi にある WSGIServer を用い、その上で flask を使い、ルーティングを行なっている。

5.1.9 メッセージングシステム

本実装では、kafka をメッセージングシステムとして用いている。kafka は consumer と broker, producer からなり、各々がデータの受け手、コーディネイター、送り手である。各々でクラスターが構成可能で、可用性に優れたメッセージングシステムであるため、本研究にて用いた、バージョンであるが、この kafka 内における SASL_PLAIN 認証を用いてユーザ認証を行うものの、現状の最新バージョンである 2.1.0 や 2.0.0 ではこの認証が動作しなかった。したがってメジャーダウングレードを行い、1.0.2 のバージョンを用いている。また、scala は 2.12 のバージョンを用いている。そしてユーザ認証は以下の conf ファイルを用いた SASL 認証を使用している。

ソースコード **5.7:** SASL 認証の設定 config/producer.properties

```
sasl.mechanism=PLAIN
security.protocol=SASL_PLAINTEXT
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \
  username="yoshiyuki" \
  password="yoshiyuki-secret";
```

また、JVM を立ち上げた後でも、このファイルにユーザ記述部分を足すことで kafka へ動的にユーザの追加が出来る。これにより、新しいユーザの追加を行っている。また、シェルを経由して bin/kafka-acls.sh を用いることで動的にユーザの権限の制限が可能となる。これにより、トークンを送信してこなくなったユーザからデータを購読する権限を剥奪している。

また、kafka は CAP の定理における CA を満たすものであり、P である Partition Tolerance(分断耐性) は低いものである。これについて、kafka は本来は同一データセンター内にて用いられることが前提のプロダクトであり、分断される可能性が低いので問題ないとしている。しかし、今回のユースケースでは kafka の producer と consumer は同一データセンター内に存在しないことの方が多い。つまり、ネットワークが分断される可能性は十分に考えられるのだ。そこで acks というパラメタが設定できる。ここではレプリケーションされたメッセージがいくつ送信成功することを要求するかを指定でき、これに all を指定すると全てのレ

プリケーションを格納する broker に情報が書き込まれることを、producer が要求することとなる。さらに replication.factor によってそのメッセージをいくつの broker が持つかについて指定することができる。これらを all や大きな数字に設定を行うと、ある broker と consumer の分断が起こったとしても他のネットワークを用いてデータの配信は可能なままである。このように、kafka では A の Availability(可用性) を犠牲にすることで P の分断耐性を増すことが可能であり、今回のユースケースでは P を増すことを考慮する必要がある。

5.2 まとめ

本章では本研究にて用いるブロックチェーン技術や、それを IoT データのような短い間隔の支払いが必要な場合に対応するためのオフチェーン技術、そしてデータを多くのスループットで拡張性を保持しつつ送受信するメッセージングシステムについて述べた。

参考文献

[1] 株式会社日立製作所, 交通データ利活用サービス, http://www.hitachi.co.jp/products/it/lumada/solution/lumada_s_010044.html (参照 2018-12-13)

[2] 株式会社セラク, Thermo-Cloud, http://www.seraku.co.jp/iot-ps/thermo.php (参照 2018-12-13)

[3] EverySense,Inc., EverySense, https://every-sense.com/ (参照 2018-12-13)

[4] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", http://wfc-knowledgecentre.com/wp-content/uploads/2016/07/Bitcoin-A-Peer-to-Peer-electronic-Cash-System.pdf(参照 2018-12-15)

[5] W. Dai, "b-money", http://www.weidai.com/bmoney.txt, (参照 2018-12-15)

[6] Vitalik Buterin, "A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM", https://whitepaperdatabase.com/wp-content/uploads/2017/09/Ethereum-ETH-whitepaper.pdf, (参照 2018-12-15)

[7] Jack Peterson, Joseph Krug, Micah Zoltu, Austin K. Williams, and Stephanie Alexander, "Augur: a Decentralized Oracle and Prediction Market Platform", https://www.augur.net/whitepaper.pdf, (参照 2018-12-15)

[8] David Schwartz, Noah Youngs, Arthur Britto, "The Ripple Protocol Consensus Algorithm", https://ripple.com/files/ripple_consensus_whitepaper.pdf, (参照 2018-12-15)

[9] 目黒 克幸, 証券市場と市場監視の役割 証券市場と市場監視の役割－真の規律が効いた市場の実現を目指して－, https://www.fsa.go.jp/sesc/kouen/kouenkai/20101117-1.pdf (参照 2018-12-14)