特別研究報告

題目

# ブロックチェーンを利用したサプライチェーンにおける データプライバシー保護と追跡性の両立が可能な プロトコルの提案と動作実験

指導教員
村田 正幸 教授

報告者
上杉 太氣央

2020 年 2 月 10 日

大阪大学 基礎工学部 情報科学科

令和元年度 特別研究報告

# ブロックチェーンを利用したサプライチェーンにおけるデータプライバシー保護と追跡性の両立が可能なプロトコルの提案と動作実験

上杉 太氣央

## 内容梗概

近年、サプライチェーンのグローバル化に伴い、偽造品の流通拡大や、問題が発生した製品の所在特定に要する時間の増大など、様々な問題が顕在化してきている。これらの問題を解決するために、サプライチェーンにおける追跡性を高い水準で担保することが急務の課題である。課題解決のアプローチとして、ブロックチェーンを用いて製品情報を管理する手法が提案されている。これら既存手法では、ブロックチェーンをサプライチェーン情報を管理するための共有データベースとして利用することで、サプライチェーンにおける流通情報を統合管理する。ブロックチェーンは透明性と改ざん耐性を有することから、流通情報の真正性が担保される。こうした特性から、サプライチェーンにブロックチェーンを適用することで追跡性、すなわち流通における製品の所在を特定できることを担保できる。しかし一方で、ブロックチェーンに記録されている情報は誰でも閲覧可能であるため、データプライバシーが保護されない。サプライチェーンにおいては、競争優位を築いている企業間の流通情報や、二次流通市場における個人間の流通情報など、プライバシー性の高い情報までもが公開されてしまう。そこで本報告では、流通情報を暗号化によって隠蔽することで事業者のデータプライバシーを保護する方法を提案する。また不正な流通を防ぐために、事業者がゼロ知識証明を利用することで、ある製品の真正な流通業者であることを証明しつつも、事業者の特定に繋がる情報を知られないようにする方法も提案する。提案手法が十分な追跡性を担保できることを確かめるために、Ethereum のスマートコントラクトを用いて提案手法を実装し、様々な製品流通のシナリオに基づいて実動作の確認を行った。その結果、流通情報を隠蔽できること、不正な流通を防げること、製造者による製品の追跡および製品の所在特定ができることを確認した。

## 主な用語

サプライチェーン、ブロックチェーン、プライバシー保護、追跡性、ゼロ知識証明

# 目 次

# 図 目 次

# 表 目 次

# 1　はじめに

サプライチェーンとは、製品の原材料・部品の調達から、製造、在庫管理、配送、販売、消費までの全体の一連の流れのことを指す。近年、サプライチェーンのグローバル化が急速に進んでいることに起因し、特に製品の追跡性に関して深刻な問題が発生してきている。例えば、偽造品の流通額は 2013 年に 4610 億米ドルだったのに対し、2016 年には 5090 億米ドルまで増加している [1]。これは、流通経路や流通情報管理が複雑化することによって、製品の製造者の真正性を確認することが困難になってきているためであると考えられる。また、食材の流通を追跡できていなかったことにより、食中毒を引き起こしている食材の所在を特定できず、食中毒の被害がアメリカの 14 州に渡り計 60 人にまで拡大してしまった事例もある [2]。そのため、サプライチェーンにおける製品の追跡性を高い水準で担保することが急務の課題である。

しかし、現状のサプライチェーンのデータ管理方法では、高い追跡性を保証することが困難である。なぜなら、サプライチェーンに参加している各事業者は、図 1 のように製品の流通情報を独自形式で個別に管理しているためである。よって、同じ製品であっても、各事業者が異なる識別子やデータ構造を用いて管理していることがありうる。そのため、製品の流通過程を追うために、事業者をまたいだ管理情報のデータクレンジングが必要になり、流通経路がわかるまでに多くの時間が必要になる。また、流通情報を偽装することにより、ブランド品や産地の偽造などが容易に実行可能であることが指摘されている [3]。

そこで、サプライチェーンにおける製品の追跡性の向上を目的として、図 2 に示すように、ブロックチェーンをサプライチェーン情報を管理するための共有データベースとして利用することによって、流通情報を統合管理し、追跡性を担保する手法が提案されている [4–11]。ここで流通情報とは、製品の所有者の移転を記録した情報である。これら既存手法では、事業者とブロックチェーン上のアドレスは一意に紐付いていること仮定する。図 2 では、アドレス `0x14723A09AC` は事業者 $M$ であることが分かっている。またブロックチェーン上でプログラムを実行する仕組みであるスマートコントラクトを活用することで、流通情報の登録に適切な条件を設けることによって、不正な流通情報を登録することができなくなる。さらに、ブロックチェーンの高い改ざん耐性により、ブロックチェーンに一度保存された流通情報は改ざんできない。そのため、ブロックチェーンを用いて流通情報を管理することで、高い追跡性を確保することができる。

しかし一方で、ブロックチェーンを活用することで新たな課題が発生する。その一つにデータプライバシーの問題が指摘されている [4]。ブロックチェーンは高い透明性を備えているため、ブロックチェーンに保存されている製品の流通情報は誰でも自由に閲覧できる。そのため、ブロックチェーンをサプライチェーンに活用したシステムでは、秘匿にしておき

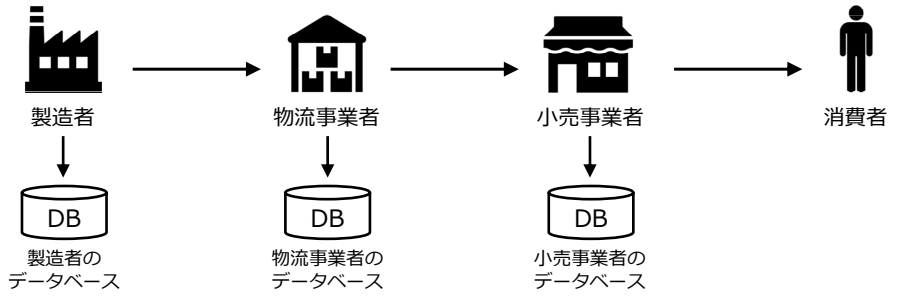


図 1: 現状のサプライチェーンにおけるデータ管理

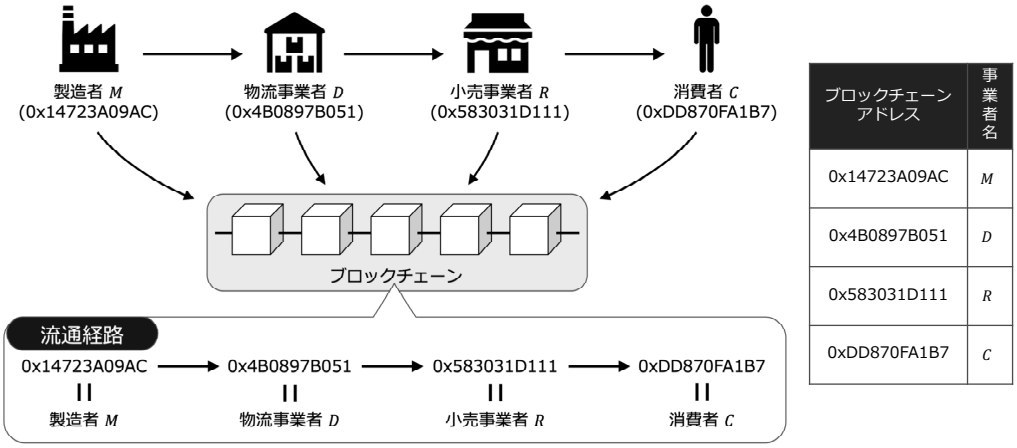


図 2: ブロックチェーンを利用したサプライチェーンのデータ管理

たい情報までもが公開されてしまう。例えば事業者は、仕入価格を下げたり迅速な流通を実現したりするために、多くの費用を投資して取引先の調査および流通関係の構築を行っている。この情報が公開されてしまうと、競合他社は投資なしに流通関係の把握・構築が可能であるため、事業者の競争上の優位性を脅かすこととなる。また昨今では、フリーマーケットアプリなどを利用して個人間取引が容易に実行できることから、二次流通市場も大きな盛り上がりを見せている。文献 [4] では、ブロックチェーンを活用した流通情報管理を導入することで二次流通市場における偽造品等の流通を防ぐ方法も提案しているが、この方法では個人間の取引情報や、製品の所有者情報が特定可能になってしまう。そのため、流通情報はプライバシー情報であり、秘匿するべき情報といえる。

そこで、本報告では、ブロックチェーンを利用したサプライチェーン管理の既存手法を拡張することで、追跡性を担保しつつも、事業者のデータプライバシーを保護可能なプロトコルを提案する。本報告では、流通に関与する個人・法人を総称して「事業者」、製品を製造して流通の起点となる事業者を「製造者」と呼ぶ。既存手法においてデータプライバシーが

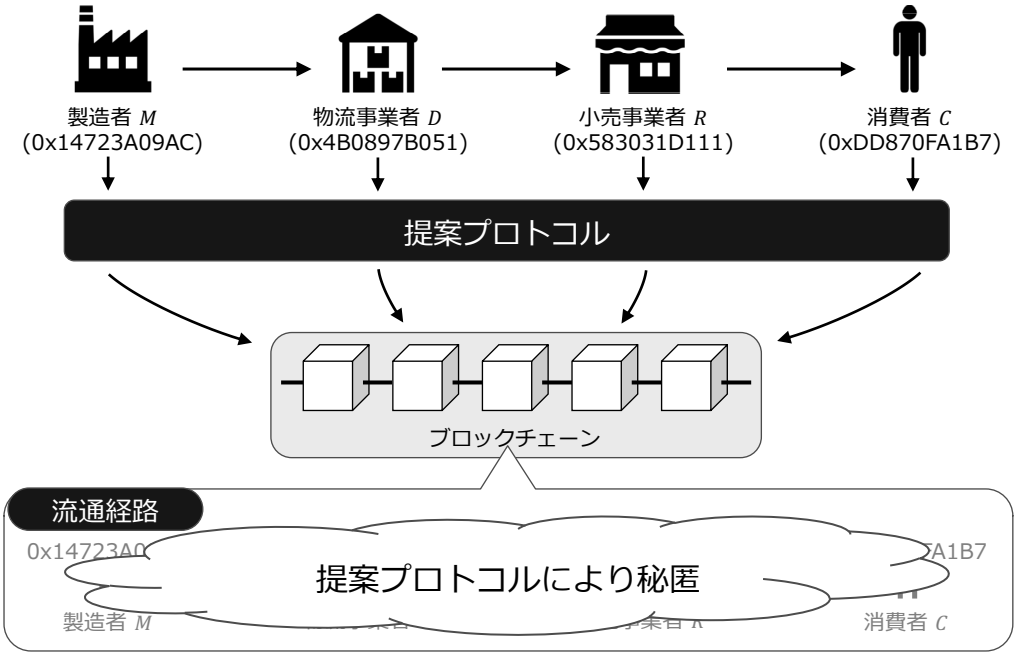


図 3: 提案手法によるブロックチェーンを活用したサプライチェーンのデータ管理

保護されていない要因は、事業者を示すブロックチェーンのアドレスが流通情報に含まれていることにある。そこで、図 3 に示すように、このアドレスを暗号化によって隠蔽することで、事業者のデータプライバシーを保護する手法を提案する。

なお、本報告における「追跡性」を、次の全ての要件を満たすことと定義する。

- 製品が、真正な製造者が製造したものであること (偽造品でないこと) を確認できる。
- 製品が、真正な事業者のみを経由して流通されていることが保証できる。
- 製品の問題発生時には、製造者が製品の所在を特定することで被害の拡散を抑えることができる。

これらの要件が満たされると、偽造品の流通の防止、不正な流通経路の排除、製品所在の特定が可能になる。よって製品流通を十分に追跡できている。

追跡性とデータプライバシーを両立するために、本報告では次の要件を満たす手法を提案する。

- 製品の流通は、その製品の真正な製造者のみが開始できる。
- 事業者は、その製品の製造者を知ることができる。

- 製品の発送は、その時点における製品の所有者のみが実行できる。発送時には、製品の受領者を指定できる。
- 製品の受領は、指定された受領者のみが行うことができる。受領と同時に、受領者が製品の新たな所有者として登録される。
- 製品の真正な製造者は、その製品の全ての流通情報を知ることができる。
- 製品の流通に関与しない事業者は、製品の製造者以外の一切の流通情報を知ることができない。
- 製品の流通に関与する事業者は、製品の製造者と、製品の発送元である発送者および製品の発送先である受領者以外の一切の流通情報を知ることができない。

事業者は、製品の発送時には自身が真正な所有者であることを、受領時には自身が真正な受領者であることを示す必要がある。そこで、事業者がゼロ知識証明を利用することで、事業者の特定に繋がるブロックチェーンアドレスを隠蔽したまま自身の真正性を証明する方法も提案する。また提案手法が、これらの要件を満たすことを、シナリオを用いた動作実験により示す。

なお、提案するプロトコルでは、単一製品が形を変えずに流通することを想定し、複数の製品（原材料）から新たな製品が作成されることは考慮しない。また、製品の追跡は製造者のみが可能であり、サプライチェーンの下流に位置する消費者から、上流の製造者に向けての製品の追跡はできない。

# 2　関連研究

本章では、本報告に関連した研究として、ブロックチェーン、ブロックチェーンを利用したサプライチェーン管理システムおよびゼロ知識証明について説明する。

## 2.1　ブロックチェーン

### 2.1.1　ビットコインとブロックチェーン

ビットコイン [12] は 2009 年に稼働を開始した世界初の暗号資産である。ビットコインは中央管理者を必要とせずに、金銭的な価値のあるデータを移転できるシステムであることから、大きな注目を集めた。このシステムを実現するために提案された手法が、分散台帳技術のブロックチェーンである。

ブロックチェーンは、ブロックと呼ばれるデータの単位が、一つ前のブロックのハッシュ値を持つことで、チェーンのように連結するデータ構造をとる。ブロックチェーンのブロックは、図 4 のように、ブロックヘッダとトランザクションデータから構成される。ブロックヘッダは、ハッシュ値、ナンス値、タイムスタンプから構成される。ハッシュ値には、1 つ前のブロックヘッダのハッシュ値を記録する。これによりチェーン構造を実現する。タイムスタンプには、ブロックが生成された時刻を記録する。ナンス値はコンセンサスアルゴリズムの実行に必要な値である。詳細は後述する。このようにチェーン構造を持つ仕組みから、仮にブロックチェーンネットワークに攻撃者が存在し、その攻撃者がトランザクションを変更しようとしても、そのトランザクションを含むブロックのハッシュ値が変わることでチェーン構造が途切れてしまうことから、容易にトランザクションの改ざんを検知できる。

トランザクションはブロックチェーンにおけるデータの最小単位である。ビットコインの場合は、トランザクションとして資産の取引履歴を表現する。トランザクションにはトランザクション発行者による電子署名が付与されることから、トランザクション実行者の確認と否認防止が実現できる。また、ブロックチェーンは、図 5 に記すように、P2P ネットワークを通じて非中央集権的に管理され、ブロックチェーンのブロック情報はブロックチェーンネットワークに参加する全てのノードで同じものを共有する。そのため、ゼロダウンタイム稼働および半永久稼働といった特徴を持つ。

ブロックチェーンでは、ネットワーク全体でただ 1 つのブロックチェーン情報を同期するための仕組みが必要である。この仕組みを実現するためのアルゴリズムをコンセンサスアルゴリズムといい、ビットコインでは PoW(Proof of Work) というアルゴリズムを採用している。PoW では、一つ前のブロックのハッシュ値、ブロックが内包するトランザクション、ナンス値を入力とした暗号学的ハッシュ値が、特定の値以下になるようなナンス値を求めると

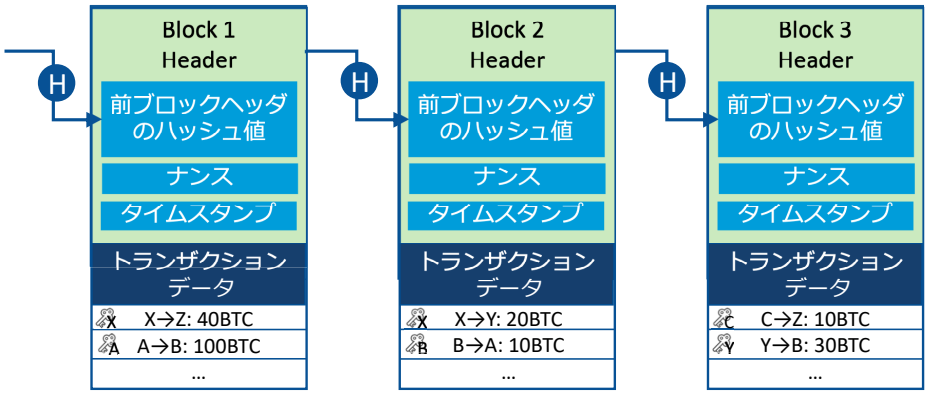


図 4: ブロックチェーンのブロック情報

いうものである。この特定の値は、ネットワークに参加する全てのノードが同時にハッシュ値を計算する環境下で、およそ 10 分に一度の頻度で成功するように自動的に調整される。暗号学的ハッシュ値の予測不可能性から、ブロックを生成するためには、ナンス値を小刻みに変更しつつハッシュ値の計算を繰り返す必要がある。よって、ブロックを生成するためには膨大な計算量が必要となる。さらに、ビットコインでは最も長いブロックチェーンを採用するというルールになっているため、攻撃者がトランザクションを改ざんするためには、その時点から現在に至るまでの全てのブロックについて再度計算し直す必要がある。これは、攻撃者が圧倒的な計算量を保持していなければ不可能であるが、このような攻撃者は現実的には存在し得ない。こうした仕組みが、ブロックチェーンの改ざん耐性を保証している。

### 2.1.2 チューリング完全な計算実行環境を備えたブロックチェーン

ビットコインはスタックベースのスクリプト言語を備えているが、チューリング完全ではない。そこで、より複雑かつ汎用的な処理を行うことを目的に、チューリング完全な計算実行環境を備えたブロックチェーンが登場した。このようなブロックチェーンはブロックチェーン上で任意のプログラムを実行可能であり、この仕組みはスマートコントラクトと呼ばれる。

スマートコントラクトが利用できるブロックチェーンで代表的なものとして、Ethereum [13] がある。Ethereum には、2 種類のアカウントが存在する。一方のアカウントである EOA (Externally Owned Accounts) は、秘密鍵から求めた公開鍵のハッシュ値から取得できる値をアカウントを表すアドレスとして使用する。秘密鍵を用いた署名を付与することでトランザクションを発行することができ、このトランザクションは、Ethereum の基軸通貨である ETH を送金するために利用される他、スマートコントラクトを実行するためのメッセージ

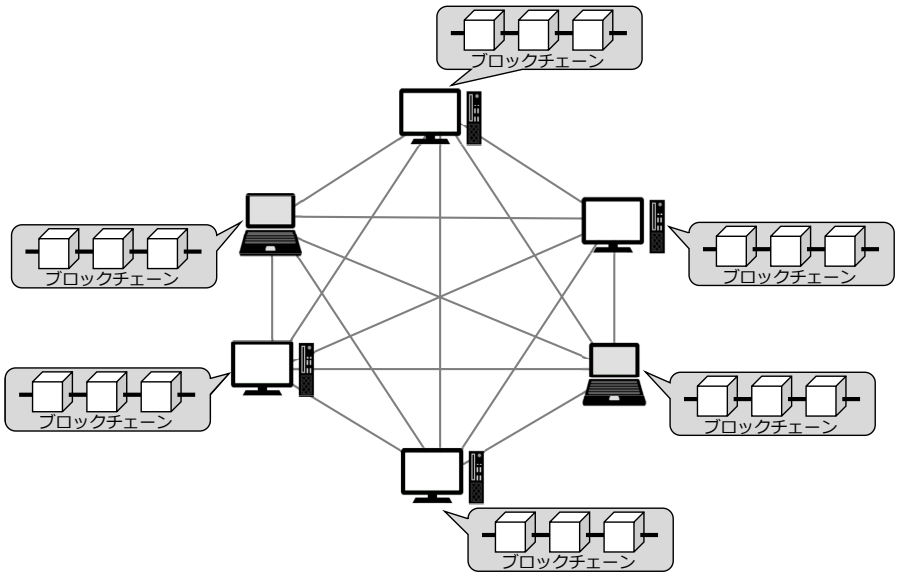


図 5: ブロックチェーンのネットワーク構成

送信としても利用される。もう一方のアカウントである CA(Contract Accounts) は、任意のプログラムとストレージを管理している。別の CA もしくは EOA からメッセージを受信した際、メッセージの内容を引数として、プログラム中の特定の関数を実行する。プログラムの実行結果は、CA のストレージに保存される、もしくは、他の CA へのメッセージとして利用される。このアカウントはブロックチェーンにより管理されるため、プログラムを非中央集権的に実行することが可能である。

本報告では、提案する手法を実現するために、Ethereum のスマートコントラクトを利用する。

## 2.2 ブロックチェーンを利用したサプライチェーン管理システム

### 2.2.1 システムの概要

サプライチェーンにおける製品の追跡性の向上を目的として、ブロックチェーンを利用したサプライチェーン管理システムが多く提案されてきている。文献 [4] で提案されている手法は、製品の所有権をブロックチェーンで管理することで、製品を追跡する。文献 [5] で提案されている手法では、製品を TRU (Traceable Resource Unit) という単位で管理し、これの消費・生産を繰り返すことで、製品を追跡する。TRU の消費・生産の履歴をブロックチェーンに保存することによって、製品の追跡性を担保している。また、実際にブロックチェーンを利用したサプライチェーン管理システムを提供し、実稼働させている企業も存在

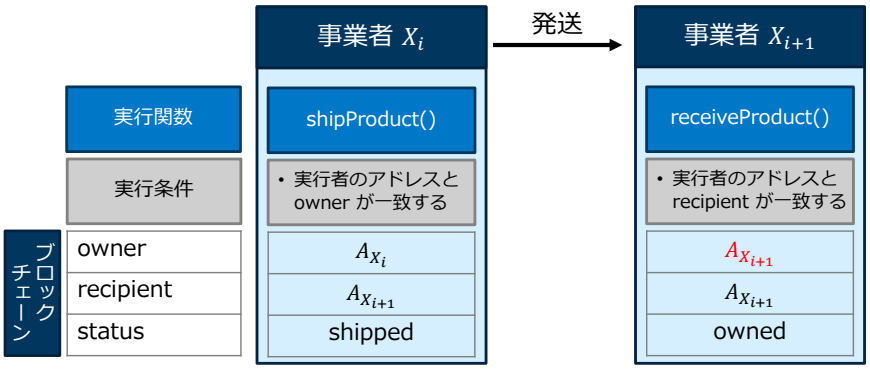


図 6: POMS を利用した事業者 $X_i$ による製品の発送および事業者 $X_{i+1}$ による受領方法

する [6–8]。

それぞれの手法において、製品を表現するためのデータ構造や、製品の所在を特定するための方法が異なっているものの、ブロックチェーン上に製品の流通記録を保存することによって、製品の追跡性を保証するという点では共通している。

### 2.2.2 Product Ownership Management System (POMS)

ブロックチェーンを利用したサプライチェーン管理システムをより詳細に説明するために、ここでは文献 [4] の POMS(Product Ownership Management System) を取り上げ、具体的に解説する。POMS は、製品の所有者の変遷をブロックチェーンに記録することで製品流通を管理している。製品の所有者はブロックチェーンのアドレスで管理され、所有者の変遷は主に、製品の発送および受領により管理される。ブロックチェーンとして Etheruem を利用している。

POMS を利用した、事業者 $X_i$ による製品の発送および事業者 $X_{i+1}$ による受領方法の概要を、図 6 に記す。図 6 において、$A_{X_i}$ は事業者 $X_i$ のブロックチェーンアドレスを意味する。一連の処理は、スマートコントラクトで実装されている。製品の発送処理は、関数 shipProduct() を利用して行い、受領者 (recipient) である $X_{i+1}$ をブロックチェーンのアドレス $A_{X_{i+1}}$ を用いて指定する。同時に、製品の状況 (status) を配送中 (shipped) に変更する。なお、shipProduct() は、その時点における製品の所有者 (owner) として記録されているブロックチェーンアドレス $A_{X_i}$ に対応する事業者 $X_i$ のみが実行可能である。スマートコントラクトでは、処理の実行者をブロックチェーンのアドレスとして取得できる。そのため、処理の実行者のアドレスと、現在の所有者が一致すれば、正しい所有者からの処理要求と見なし、発送処理を実行する。受領処理は、関数 receiveProduct() を利用して行う。正しい受領

者からの処理要求である時のみ処理を実行し、$X_{i+1}$ を新たな所有者として記録し、製品状況を所有中 (owned) に変更する。

ブロックチェーンのアドレスは単なる数値であり、社名や住所などの実世界の身元情報とは関連付けられていない。これでは製品の追跡が困難であるため、実際は、それらを関連付けるための仕組みを備えているものと想定できる。つまり、ブロックチェーンアドレスを用いた流通管理では、流通情報が広く公開されてしまう。そのため、POMS では流通情報に関するデータプライバシーが担保されていない。他の多くの手法もブロックチェーンアドレスを用いた流通管理を行っていることから、同様にデータプライバシーの問題を抱えている。

## 2.3 ゼロ知識証明

ゼロ知識証明は、ある人（証明者）が別の人（検証者）に対して、「自身がその情報を知っている」ということを、それ以外の情報を検証者に与えずに、証明するための手法である [14]。

この証明には、完全性、健全性、ゼロ知識性という 3 つの性質が満たされていなければならない。完全性とは、証明者が証明した内容が真であるならば、検証者はそれが真であると判断できることである。健全性とは、証明者が証明した内容が偽であるならば、検証者は高い確率でそれが偽であると判断できることである。ゼロ知識性とは、検証者は、証明者の証明から「その情報が真実である」という以外の情報を得ることができないことである。

この手法を、ブロックチェーンを利用したサプライチェーン管理システムに適用することで、製品の所有者や受領者（証明者）が、ブロックチェーンネットワークの参加者（検証者）に対して、所有者・受領者の特定に繋がるような情報を与えずに、確かに所有者・受領者が自分である、ということを証明することが可能になる。

# 3　ブロックチェーンを利用したプライバシー保護と追跡性を両立した製品流通手法

本報告では、ブロックチェーンを利用した製品流通管理において、事業者の流通情報に関するデータプライバシーを保護しつつ製品の追跡性を担保する手法を提案する。本報告は、文献 [4] で提案されている Product Ownership Management System (POMS) を改良することで、手法の提案を行う。

POMS では、製品の所有者の変遷をブロックチェーンのアドレスで管理する。所有者の変遷を追跡するために、ブロックチェーンアドレスと社名や住所などの実世界の身元情報とは関連付けるための別の仕組みが用意されていると想定できることから、このままでは誰でも自由に流通情報を取得することができるため、プライバシーが保護されない。

そこで、本報告では、製品の所有者として記録されるブロックチェーンアドレスを隠蔽することで、プライバシーを保護する。しかし、ここで問題となるのが製品の所有者あるいは受領者であることを証明する方法である。製品の追跡性を保証するためにも、製品を発送する際には、その製品の所有者であることを証明する必要があり、製品を受領する際には、その製品の受領者であることを証明する必要がある。POMS では、製品の所有者や受領者の情報はブロックチェーンのアドレスで管理されていたため、製品の発送処理、受領処理の実行者のブロックチェーンアドレスと比較することによって、正しい所有者・受領者であることを容易に確認することができた。しかし、ブロックチェーンアドレスを隠蔽した場合、処理の実行者のブロックチェーンアドレスと比較を行うことによる確認ができなくなってしまう。そこで、本報告では、ゼロ知識証明を用いて、この問題を解決する。ゼロ知識証明を利用することで、製品の所有者や受領者が、ブロックチェーンの情報を閲覧可能な全ての人に対し、「自身は製品の正しい所有者あるいは受領者である」という証明を、その証明の正誤以外の一切の情報を与えることなく示すことが可能になる。この証明が正しいものであると検証されることで、製品の正しい所有者あるいは受領者であると判断されたことになるため、製品の発送処理あるいは受領処理の実行が許可される。こうすることで、製品の所有者・受領者のブロックチェーンアドレスを隠蔽することでプライバシーを担保しつつも、追跡性を保証した製品の流通を実現できる。

以降、提案する手法が想定するシステムモデルについて説明し、プロトコルの全体概要について述べる。その後、ブロックチェーンアドレスの隠蔽方法およびゼロ知識証明を用いた所有者・受領者であることの証明方法について解説し、プロトコルの詳細について説明する。最後に、製造者による製品の追跡方法について説明する。

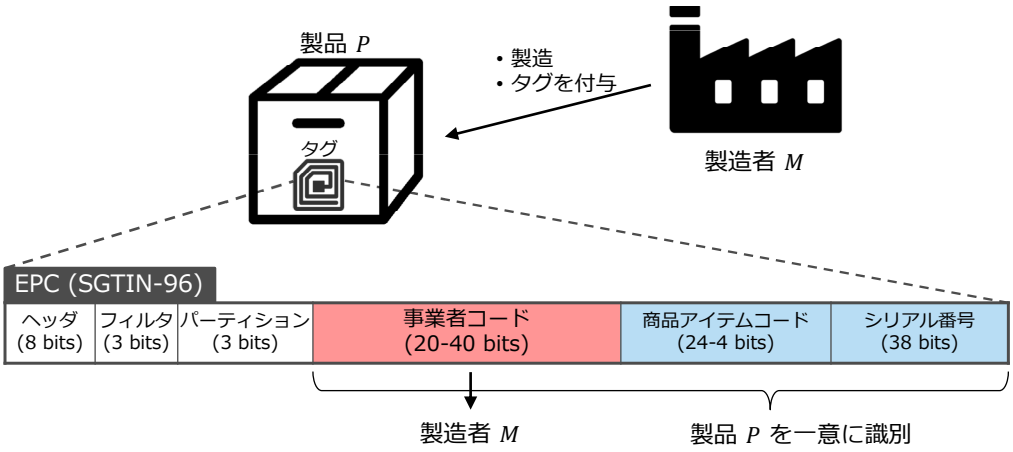


図 7: 製品に付与されるタグに記録される情報

### 3.1 システムモデル

提案する手法では、次のシステムを想定する。

製品には、図 7 に示すように、RFID や QR コードなどの製品を管理するためのタグが付与されており、そこには EPC(Electronic Product Code) が書き込まれている。EPC は、GS1 という国際標準化団体により標準化された規格であり、製品とその製造者を識別するために利用する。EPC には SGTIN-96 という個別識別コードを使用する。SGTIN-96 は製品の製造者を一意に識別可能な事業者コードおよび製品を一意に識別可能なシリアル番号を含んでいる。提案する手法では、この EPC を用いて製品情報を管理する。なお、RFID と QR コードを付け替える・改ざんするなどにより、製品に付与された EPC 情報が変更されることは無いものとする。

提案する手法では、ブロックチェーンネットワークにブロックチェーンプラットフォームの Ethereum を使用する。事業者は、Ethereum で有効な秘密鍵・公開鍵のペアを所持する。秘密鍵は、その所有者と 1:1 の関係にあり、他者に知られないように厳重に管理する。公開鍵は、第三者が自由に取得することができる。公開鍵のハッシュ値として求められるブロックチェーンアドレスは、事業者との対応がわかっているものと仮定する。また、他者は、公開鍵やブロックチェーンアドレスから、元の秘密鍵を復元することはできない。

### 3.2 プロトコルの全体概要

ここでは、プロトコルの全体概要を製品流通の具体例を用いて説明する。例として、製造者 $M$、事業者 $X_i$、事業者 $X_{i+1}$ の順で製品を流通することを想定する。

表 1: 流通管理のためのブロックチェーンに記録される製品情報

| 変数名 | 管理する情報 |
|---|---|
| manufacturer | 製品の製造者 |
| owner | 製品の所有者 |
| recipient | 製品の受領者 |
| status | 製品の状態 |
| cntAddr | 証明の検証用スマートコントラクトのアドレス |
| proof | 製品受領時の証明 |

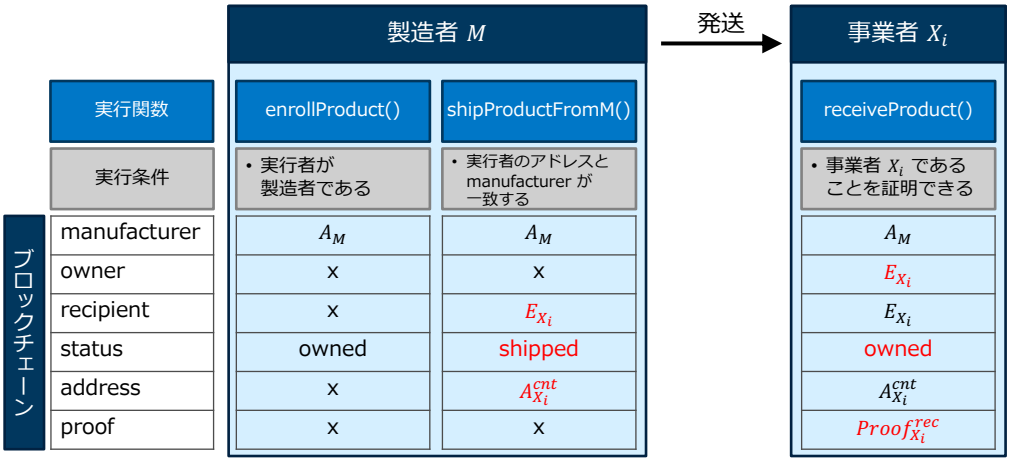


図 8: 製造者 $M$ から事業者 $X_i$ へ製品を流通させる際にブロックチェーンに記録される情報

まず、流通管理のためにブロックチェーンに記録される情報を表 1 に示す。これらの情報は製品に付与されている EPC に一意に対応づけられて、ブロックチェーンに記録される。以降は、表 1 の情報を示しながら、流通の流れを説明する。

図 8 は製造者 $M$ から事業者 $X_i$ へ製品を流通させるときに、実行される処理およびブロックチェーンに記録される情報をまとめたものである。この図をもとに、製造者 $M$ から事業者 $X_i$ への流通を説明する。製品流通の際には、ブロックチェーンに登録されている製品情報を管理するため、スマートコントラクト ProductsManager が提供する関数を実行することで、ブロックチェーンに記録されている製品情報を変更する。以降、関数は ProductsManager が提供していることとする。

まず、製造者 $M$ は製品を製造すると enrollProduct() を実行して、製品の EPC をブロックチェーンに登録する。この際、EPC が示す事業者コードが製造者 $M$ の事業者コードと一致しなければならない。これにより、不正な製造者による製品登録を防ぐ。登録する情報と

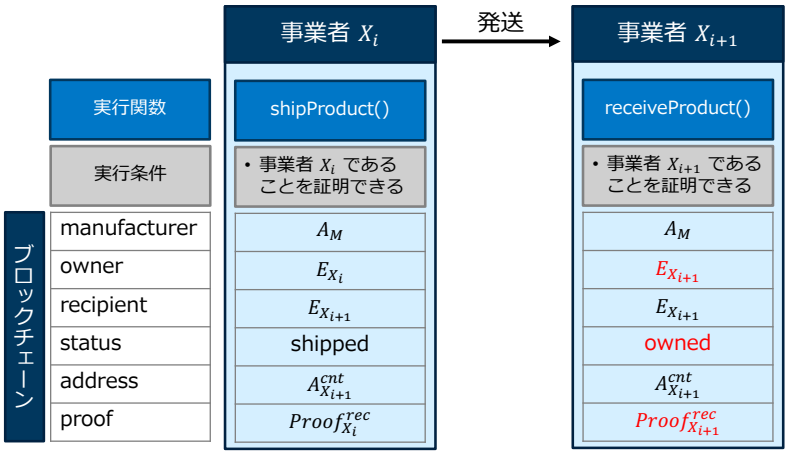


図 9: 製造者 $X_i$ から事業者 $X_{i+1}$ へ製品を流通させる際にブロックチェーンに記録される情報

しては、製造者を表す変数 manufacturer に製造者 $M$ のブロックチェーンのアドレスである $A_M$ を記録し、製品の状態を表す変数 status に製品が所有されている状態にあることを示す owned を設定する。

次に、製造者 $M$ は事業者 $X_i$ に製品を発送するために shipProductFromManufacturer() (図 8 では shipProductFromM() と記載) を実行する。ここでは、製品の受領者を表す変数 recipient に事業者 $X_i$ を表す数値 $E_{X_i}$ を指定する。ただし、発送者である製造者 $M$ と受領者である事業者 $X_i$ のみが、この数値は事業者 $X_i$ を表すものであることがわかる。$E_{X_i}$ の生成方法は 3.3 節で説明する。加えて、事業者 $X_i$ が受領・発送する際の証明を検証するスマートコントラクトのアドレス $A^{cnt}_{X_i}$ を変数 cntAddr に記録する。また、製品の状態を表す変数 status に製品が発送されている状態にあることを示す shipped を設定する。

そして、事業者 $X_i$ は製品を受領するために receiveProduct() を実行する。ここで、事業者 $X_i$ は自身が確かに事業者 $X_i$ であることを証明するための証明情報 $Proof^{rec}_{X_i}$ を生成する。証明情報はアドレス $A^{cnt}_{X_i}$ が示すスマートコントラクトにより検証される。検証結果が真、つまり receiveProduct() の実行者が事業者 $X_i$ であると判断されると、製品の所有者を表す変数 owner に事業者 $X_i$ を表す $E_{X_i}$ を記録し、製品の状態を表す変数 status に製品が所有されている状態にあることを示す owned を設定する。なお、一度利用された証明情報は公開情報となる。事業者 $X_i$ が製品を発送する時には、自身が所有者であることを示す証明情報を生成し、スマートコントラクトを通じて検証される必要があるが、受領時に利用した証明情報 $Proof^{rec}_{X_i}$ はスマートコントラクトの検証を通過してしまうため、$Proof^{rec}_{X_i}$ を使い回すことで、事業者 $X_i$ 以外の別の事業者が不正に発送処理を行うことができてしまう。そのため、事業者 $X_i$ が製品を発送するときに、使用する証明が使い回されていないことを

確認するために、ここで使用された証明 $Proof_{X_i}^{rec}$ を変数 $proof$ に記録する。

次に、事業者 $X_i$ から事業者 $X_{i+1}$ への流通を説明する。図 9 は、この流通における実行される処理およびブロックチェーンに記録される情報をまとめたものである。まず、事業者 $X_i$ は事業者 $X_{i+1}$ に製品を発送するために shipProduct() を実行する。ここで、事業者 $X_i$ は自身が確かに事業者 $X_i$ であること証明するための証明情報を生成する。この証明情報と受領時に使用した証明情報 $Proof_{X_i}^{rec}$ を比較し、受領時の証明情報が流用されていないことが確認される。その後、証明情報はアドレス $A_{X_i}^{cnt}$ が示すスマートコントラクトにより検証される。検証結果が真、つまり shipProduct() の実行者が事業者 $X_i$ であると判断されると、製品の受領者を表す変数 recipient に事業者 $X_{i+1}$ を表す数値 $E_{X_{i+1}}$ を指定する。ただし、発送者である事業者 $X$ と受領者である事業者 $X_{i+1}$ のみが、この数値は事業者 $X_{i+1}$ を表すものであることがわかる。加えて、事業者 $X_{i+1}$ が受領・発送する際の証明を検証するスマートコントラクトのアドレス $A_{X_{i+1}}^{cnt}$ を変数 cntAddr に記録する。また、製品の状態を表す変数 status に製品が発送されている状態にあることを示す shipped を設定する。

そして、事業者 $X_{i+1}$ は製品を受領するために receiveProduct() を実行する。ここで、事業者 $X_{i+1}$ は自身が確かに事業者 $X_{i+1}$ であることを証明し、その証明はアドレス $A_{X_{i+1}}^{cnt}$ が示すスマートコントラクトにより検証される。事業者 $X_{i+1}$ であると判断されると、製品の所有者を表す変数 owner に事業者 $X_{i+1}$ を表す $E_{X_{i+1}}$ を記録し、製品の状態を表す変数 status に製品が所有されている状態にあることを示す owned を設定する。これに加え、事業者 $X_{i+1}$ が製品を発送するときに使用する証明が流用されていないことを確認するために、ここで使用された証明 $Proof_{X_{i+1}}^{rec}$ を変数 $proof$ に記録する。

以降は事業者 $X_i$、事業者 $X_{i+1}$ 間と同様の製品の発送と受領を繰り返すことで、製品は流通される。

### 3.3 プライバシー保護を目的としたブロックチェーンアドレスの隠蔽方法

提案手法では、製品の所有者として記録されるブロックチェーンアドレスを隠蔽することで、プライバシーを保護する。ここではその方法について説明する。

提案手法では、製品とその所有者に関連した情報を、楕円曲線暗号を用いて暗号化し、この暗号化により得られた暗号文を、製品の所有者として記録することで、ブロックチェーンアドレスを隠蔽する。具体的には、製品の EPC とその製品の発送者・受領者のブロックチェーンアドレスの排他的論理和を求め、それを製品の製造者の公開鍵で暗号化する。この暗号文を製品の所有者として記録する。

楕円曲線は次の式 (1) の形で定義される。

$$ax^2 + y^2 = 1 + dx^2y^2 \ (mod\ p) \tag{1}$$

楕円曲線暗号の安全性は、楕円曲線上の離散対数問題の困難性に依存する。楕円曲線上の離散対数問題は、使用する楕円曲線の位数が大きいほど解くのが困難になる。そのため、位数が十分に大きくなるような楕円曲線を選択、つまり式 (1) 中のパラメータ $a, d, p$ を選択する必要がある。ここでは、$a, d, p$ を適切に選択した楕円曲線を $E$ と表記する。なお、本報告の動作実験で使用する楕円曲線のパラメータは、4.1 節に記載する。

暗号文 $C$ は、楕円曲線 $E$ 上の点の組であり、次の式 (2) を用いて計算する。

$$C = (rG, T + rQ) \tag{2}$$

ここで、$r$ は自然数の乱数、$G$ は楕円曲線 $E$ の生成元、$T$ は平文、$Q = sG$ は自然数 $s$ を秘密鍵として生成される公開鍵である。なお、式 (2) 中の演算は、全て楕円曲線上における演算である。

$t$ を、製品の EPC とその製品の発送者・受領者のブロックチェーンアドレスの排他的論理和の値とすると、$t$ は必ずしも楕円曲線上の点にはならない。そのため、文献 [15] を参考に、$t$ を楕円曲線上の値へと変換し、それを平文 $T$ とする。具体的には、$t$ を 100 倍し、$i$ $(0 \leq i \leq 99)$ を加算して異なる 100 個の $x_i$ を生成する。この $x_i$ を式 (1) の $x$ に代入し、$y$ が整数、つまり楕円曲線上の点となる最小の $x_i$ を平文 $T$ の $x$ 座標とする。

### 3.4 ゼロ知識証明を用いた真正な事業者であることの証明方法

提案手法では、製品流通の事前準備として、発送者は受領者と秘密の値を共有する。この秘密の値を知っていることを証明できた者を、真正な受領者とみなす。具体的には、発送者が製品を発送する際に、受領者として記録される暗号文に使われる式 (2) の $r$ に該当する数値を、秘密の値として共有する。

そして、受領者は、この秘密の値 $r$ を知っていることを、受領者として記録されている暗号文を計算できることを証明することで、証明する。受領者として記録されている暗号文を計算するには、式 (2) の $r, G, T, Q$ に該当する数値をすべて知っている必要がある。そのため、受領者として記録されている暗号文を計算できることは、秘密の数値 $r$ を知っていることと同義といえる。この暗号文を計算できることの証明に、ゼロ知識証明を用いる。ゼロ知識証明を用いるため、証明情報から、秘密の値 $r$ や式 (2) の $T$ に該当する数値に使用されるブロックチェーンアドレス情報が公開されることはない。

次に、暗号文を計算できることという知識がゼロ知識証明における 3 つの性質である完全性、健全性、ゼロ知識性を満たすことを説明する。証明情報の作成には、暗号文を計算するための数値が必要である。これは、製品の受領前に発送者と共有しており、証明者はこれらを用いて正しい証明情報を作成することができる。そのため、完全性は満たされていると

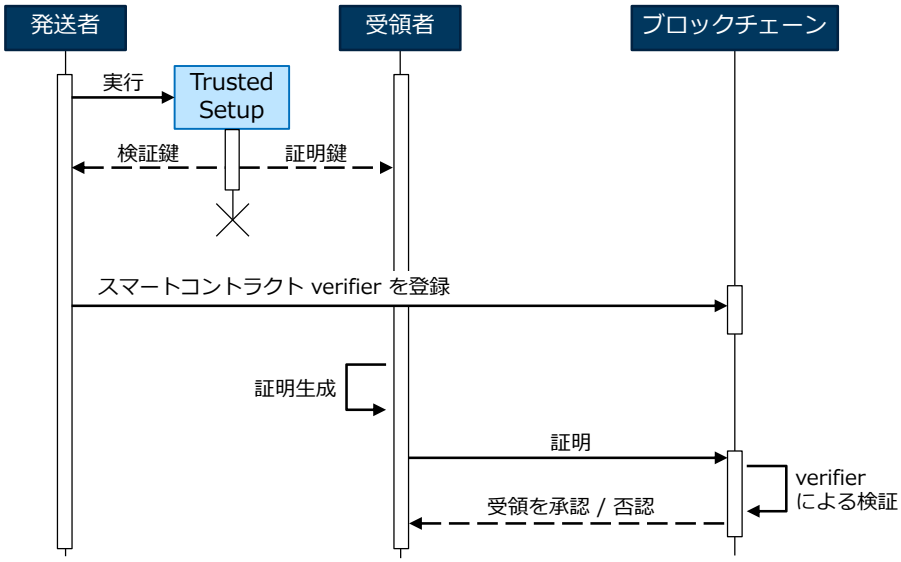


図 10: 提案手法における zk-SNARKs を利用した製品受領時の証明の検証までの流れ

いえる。また、証明情報の作成に必要となるもののうち、式 (2) に示した $r$ は、発送者と受領者の二者間でのみ共有される。そのため、第三者が証明情報を作成したとしても、正しい証明にはなりえず、検証結果は偽となる。そのため、健全性は満たされているといえる。また、証明の対象となる暗号文からは、楕円曲線上の離散対数問題の困難性から、その暗号文のもととなった数値を知ることは不可能である。そのため、ゼロ知識性は満たされているといえる。証明情報の完全性、健全性、ゼロ知識性については、4.3.1 節にて記載する。

ゼロ知識証明の手法は複数あるが、提案手法では非対話型ゼロ知識証明であり、証明サイズ、実行時間、計算量の観点からブロックチェーンにて広く使用されている zk-SNARKs というゼロ知識証明の手法を用いる。zk-SNARKs では、$f(x) = y$ の形で証明する知識を指定できる。$x$ は証明者が秘匿にしたまま真実であることを証明したい情報であり、関数 $f$ に $x$ を与えた結果の証明情報を作成する。この証明情報と $y$ を用いて、$f(x) = y$ を満たしているか検証する。提案手法において、関数 $f$ は楕円曲線暗号の暗号文を計算する関数、つまり式 (2) を計算する関数であり、$y$ は製品の受領者としてブロックチェーンに記録されている暗号文である。そして、$x$ は先に記載した暗号文の計算に必要な情報である。zk-SNARKs では、証明者と検証者の間で Trusted Setup と呼ばれる証明の事前準備が必要であり、ここで証明鍵と検証鍵が生成される。Trusted Setup には、入力として、関数 $f$ を与える必要があり、証明鍵・検証鍵は関数 $f$ に依存する。そのため、証明鍵・検証鍵を他の知識の証明に流用することはできない。証明者は証明鍵を使用して証明を作成する。検証者は検証鍵を使用して、証明者が作成した証明を検証する。

図 10 に、提案手法における zk-SNARKs を利用した製品受領時の証明における検証まで

の流れを示す。Trusted Setup は発送者が実行し、証明鍵と検証鍵を生成する。また発送者は、証明が与えられるとこの検証鍵により証明の検証がされる関数 verifyTx() を含むスマートコントラクト verifier をブロックチェーンに登録しておく。このスマートコントラクトのアドレスが表 1 に示した変数 cntAddr に記録されるものである。証明者である受領者は、このスマートコントラクト内の関数 verifyTx() に証明を与える。スマートコントラクトの仕組みにより、与えられた証明は非中央集権的に検証される。そして、この証明が正しければ、受領者は自身が正規の受領者であることを証明でき、製品を受領することができる。

いずれの説明も、製品の受領時における受領者であることの証明に関するものであったが、製品の発送時には所有者であることを証明する必要がある。ここでは、所有者として記録されている暗号文を計算できることを証明する。この所有者として記録されている暗号文は、製品を受領する際に受領者として記録されていた暗号文と同じものである。つまり、受領時と同じ数値、同じ証明鍵を使用して、証明情報を作成することができる。zk-SNARKs において、証明情報の作成には乱数が用いられており、同じ証明鍵を用いたとしても、作成される証明情報は異なるものとなる。そのため、証明の検証も受領時と同じスマートコントラクトで行うことができる。ただし、製品の発送時には、使用した証明が受領時の証明を使い回していないことを確認する。

## 3.5 プロトコルの詳細

ここでは、提案手法を構成するプロトコルの詳細を説明する。製品流通を管理する際に必要な処理は大きく分けて、製造者の登録、製品の登録、製品の発送、製品の受領の 4 つである。提案手法は、文献 [4] を参考にしたものであり、プライバシー保護に直結しない処理である製造者の登録、製品の登録は文献 [4] と同じ処理を行うものとする。以降、提案手法においてブロックチェーンに記録する情報について説明する。その後、製品流通を管理するために必要な処理である、製造者の登録、製品の登録、製品の発送、製品の受領について説明する。

### 3.5.1 ブロックチェーンに記録する情報

提案手法では、製造者と製品の情報をブロックチェーンに記録する。

#### 製造者の情報

製品を流通させるにあたり、製品をブロックチェーンに登録する必要がある。不正な製品の登録を防止するために、製品登録は、製造者のみ可能であり、自身が製造した製品のみ登録可能である必要がある。そのために必要な製造者の情報を表 2 に示す。製造者のブロック

表 2: 製造者情報を管理するための変数

| 変数名 | 説明 |
|---|---|
| companyPrefix | EPC に記録されている製造者の事業者コードを管理する |
| companyName | 製造者の名前を管理する |

表 3: 製品情報を管理するための変数

| 変数名 | 説明 |
|---|---|
| manufacturer | 製品の製造者のブロックチェーンアドレスを管理する |
| owner | 製品の所有者を表す暗号文を管理する |
| recipient | 製品の受領者を表す暗号文を管理する |
| status | 製品の状態を管理する |
| cntAddr | 証明を検証するスマートコントラクトのアドレスを管理する |
| proof | 製品受領時の証明を管理する |

チェーンアドレスに対して表 2 の情報が一意に対応づけられて、ブロックチェーンに記録される。表 2 に示される情報は、ManufacturersManager というスマートコントラクトで管理されており、このスマートコントラクトが提供する関数を実行することで、これら情報を変更する。

変数 companyPrefix、変数 companyName は、製造者をブロックチェーンに登録する際に記録され、以降変更されることはない。変数 companyPrefix は、製品をブロックチェーンに登録する際に、その製品に付与されている EPC の事業者コードが、登録しようとした製造者のものであるか確認するために使用される。これにより、自身が正規の製造者ではない製品をブロックチェーンに登録することはできなくなる。

以降では、ブロックチェーンに記録されるブロックチェーンアドレス $A_M$ の製造者の事業者コードは、manufacturers[$A_M$].companyPrefix で表示する。変数 companyName についても同様である。

### 製品の情報

提案手法において、製品流通を管理するために必要な製品の情報を表 3 に示す。提案手法では、EPC を用いて製品情報を管理するため、表 3 の情報が EPC に一意に対応づけられて、ブロックチェーンに記録される。表 3 に示される情報は、スマートコントラクト ProductsManager で管理されており、このスマートコントラクトが提供する関数を実行することで、これら情報を変更する。

変数 manufacturer は、製品の製造者のブロックチェーンアドレスを記録する。これは製品をブロックチェーンに登録する際に記録され、以降変更されることはない。変数 owner は、製品の所有者を表す暗号文を記録する。これは製品の所有者が移り変わるとき、つまり製品が受領された際に記録される。変数 recipient は、製品の受領者を表す暗号文を記録する。これは製品の発送がされた際に記録される。変数 status は、製品の状態を表すものであり、製品が所有されているときは owned、発送されているときは shipped で指定される。変数 cntAddr は、製品を受領・発送するときに必要となる、受領者・発送者であることの証明を検証するための関数 verifyTx() を含むスマートコントラクトのアドレスを記録する。変数 proof は、製品の受領時に使用された証明を記録する。これは製品の受領時に記録され、製品の発送時に、受領時の証明が流用されていないことを確認するために使用される。

以降では、ブロックチェーンに記録される情報について、$EPC$ の付与された製品の製造者は products[$EPC$].manufacturer で表示する。表 3 における他の変数についても同様である。

### 3.5.2 製造者の登録

**Algorithm 1** 製造者の情報を登録する enrollManufacturer()

**Input:** コントラクト実行者 ($A_{msg}$), 製造者のアドレス ($A_M$), 事業者コード ($companyPrefix$), 事業者名 ($companyName$)

1: **if** $A_{msg}$ が管理者である **then**
2: manufacturers[$A_M$].companyPrefix ← $companyPrefix$
3: manufacturers[$A_M$].companyName ← $companyName$
4: **else**
5: Do nothing
6: **end if**

アルゴリズム 1 は製品情報をブロックチェーンに登録する際に必要な製造者の情報を登録する処理を表す enrollManufacturer() である。この関数はスマートコントラクト ManufacturersManager で提供される。製造者の登録は、管理者の存在を想定し、その管理者のみが実行できる。この管理者には事業者コードを管理している GS1 が想定される。管理者が実行していることを確認するために、enrollManufacturer() の実行者が管理者と一致するか確認する。これが確認できると、事業者コード companyPrefix と事業者名 companyName を製造者のアドレス $A_M$ に対応づけて記録する。

### 3.5.3 製品の登録

**Algorithm 2** 製品を登録する enrollProduct()

**Input:** $EPC$, コントラクト実行者 ($A_{msg}$), ManufacturersManager のアドレス ($A_{MM}$)

1: **if** $EPC$ がブロックチェーンに登録されていない AND
   $A_{MM}$.manufacturers[$A_{msg}$].companyPrefix と $EPC$ の事業者コードが一致する **then**
2: products[$EPC$].manufacturer ← $A_{msg}$
3: products[$EPC$].status ← *owned*
4: **else**
5: Do nothing
6: **end if**

アルゴリズム 2 は、流通させる製品をブロックチェーンに登録する処理を表す enrollProduct() である。この関数はスマートコントラクト ProductsManager で提供される。ここでは、まず、登録しようとしている $EPC$ が既に登録されたものでないか確認する。次に、入力の $EPC$ 内の事業者コードが実行者 $A_{msg}$ の事業者コードと一致するか確認する。これが確認できると、EPC に対応する製品の製造者を $A_{msg}$ で記録し、製品の状態を *owned* に設定する。

### 3.5.4 製品の発送

提案手法では、製品の発送は大きく 2 つに分ける。一方は、製品の製造者により製品が発送される場合である。もう一方は、製造者以外の所有者により製品が発送される場合である。これら関数はいずれもスマートコントラクト ProductsManager で提供される。

**Algorithm 3** 製造者が製品を発送する際に実行する shipProductFromManufacturer()

**Input:** $EPC$, コントラクト実行者 ($A_{msg}$), 暗号文 ($E_{rec}$), コントラクトアドレス ($A_{verifier}$)

1: **if** products[$EPC$].status が *owned* である AND
   products[$EPC$].owner が登録されていない AND
   products[$EPC$].manufacturer と $A_{msg}$ が一致する **then**
2: products[$EPC$].recipient ← $E_{rec}$
3: products[$EPC$].status ← *shipped*
4: products[$EPC$].cntAddr ← $A_{verifier}$
5: **else**
6: Do nothing
7: **end if**

アルゴリズム 3 は製造者が製品を発送する処理を表す shipProductFromManufacturer() である。ここでは、まず、$EPC$ に対応する製品の状態が *owned* になっていることを確認する。次に、$EPC$ に対応する製品の所有者がまだいないことを確認する。そして、実行者が $EPC$ に対応する製品の製造者と一致するか確認する。これが確認できると、$EPC$ に対応

する製品の受領者を $E_{rec}$ で記録し、製品の状態を *shipped* に設定する。これに加え、受領者が製品を受領および受領後に発送する際に使用する関数 verifyTx() を含むスマートコントラクトのアドレス $A_{verifier}$ を記録する。

**Algorithm 4** 製造者以外の所有者が製品を発送する際に実行する shipProduct()

**Input:** $EPC$, 証明 ($Proof$), 暗号文 ($E_{rec}$), コントラクトアドレス ($A_{verifier}$)

1: **if** products[$EPC$].status が *owned* である AND
   products[$EPC$].proof と $Proof$ が一致しない AND
   products[$EPC$].cntAddr.verifyTx($Proof$) が真である **then**
2: products[$EPC$].recipient ← $E_{rec}$
3: products[$EPC$].status ← *shipped*
4: products[$EPC$].cntAddr ← $A_{verifier}$
5: **else**
6: Do nothing
7: **end if**

アルゴリズム 4 は製造者以外の所有者が製品を発送する処理を表す shipProduct() である。ここでは、まず、$EPC$ に対応する製品の状態が *owned* になっていることを確認する。次に、受領時に使用した証明が記録されている products[$EPC$].proof と証明 $Proof$ を比較し、一致しないことを確認する。そして、products[$EPC$].cntAddr が指すスマートコントラクトに含まれる関数 verifyTx() に証明 $Proof$ を与えることで、製品の所有者であることを確認する。これらが確認できると、$EPC$ に対応する製品の受領者を $E_{rec}$ で記録し、製品の状態を *shipped* に設定する。これに加え、受領者が製品を受領および受領後に発送する際に使用する関数 verifyTx() を含むスマートコントラクトのアドレス $A_{verifier}$ を記録する。

### 3.5.5 製品の受領

**Algorithm 5** 製品を受領する際に実行する receiveProduct()

**Input:** $EPC$, 証明 ($Proof$), 暗号文 ($E_{rec}$)

1: **if** products[$EPC$].status が *shipped* である AND
   products[$EPC$].recipient と $E_{rec}$ が一致する AND
   products[$EPC$].cntAddr.verifyTx($Proof$) が真である **then**
2: products[$EPC$].owner ← products[$EPC$].recipient
3: products[$EPC$].status ← *owned*
4: products[$EPC$].proof ← $Proof$
5: **else**
6: Do nothing
7: **end if**

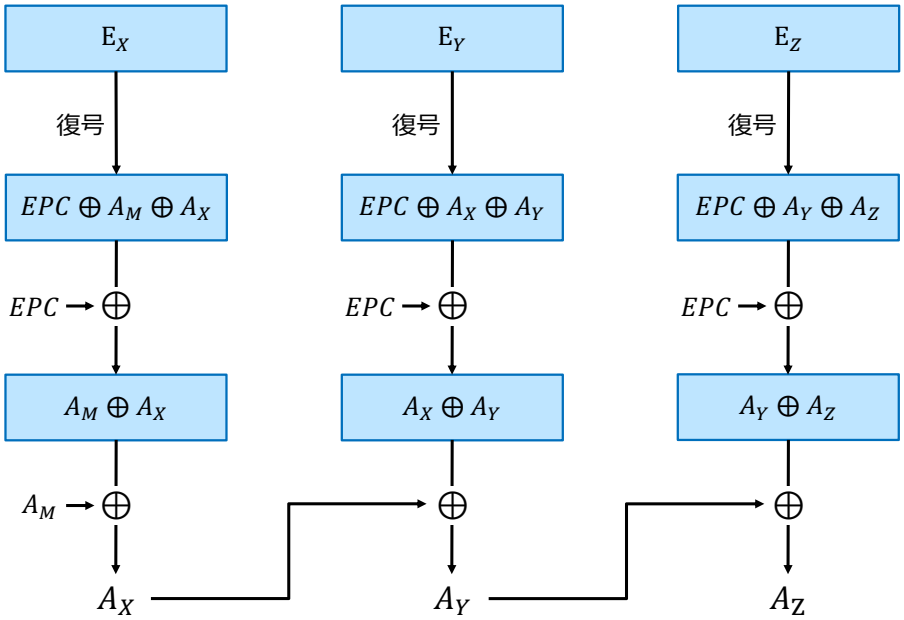


図 11: 製造者による流通経路の取得

アルゴリズム 5 は製品を受領する処理を表す receiveProduct() である。この関数はスマートコントラクト ProductsManager で提供される。ここでは、まず、$EPC$ に対応する製品の状態が *shipped* になっていることを確認する。次に、この処理の実行者は、自身を表す暗号文 $E_{rec}$ を計算し、これを与える。これが products[$EPC$].recipient と一致すれば、指定された受領者が自身であることを確認できる。また、products[$EPC$].cntAddr が指すスマートコントラクトに含まれる関数 verifyTx() に証明 $Proof$ を与えることで、製品の受領者であることを確認する。これが確認できると、$EPC$ に対応する製品の所有者を受領者に指定されていた暗号文、つまり $E_{rec}$ に変更し、製品の状態を *owned* に設定する。これに加え、使用した証明 $Proof$ を記録する。

### 3.6 製造者による製品の追跡方法

ここでは、製造者による製品の追跡方法を具体例を用いて説明する。例として、製造者 $M$、事業者 $X$、事業者 $Y$、事業者 $Z$ で製品が流通したと想定する。このとき、ブロックチェーンに記録されていた製品の所有者は $E_X$、$E_Y$、$E_Z$ である。$E_X$ は事業者 $X$ を表す暗号文であり、乱数 $r_x$ を用いて式 (3) によって計算された。

$$E_X = (r_x G, T_x + r_x Q) \tag{3}$$

$E_Y$、$E_Z$ についても、それぞれ乱数 $r_y, r_z$ を用いて同様にして計算された。このとき、図 11 に、製造者が製品の流通経路を取得する方法の概略図を示す。

この暗号文 $E_X$ は製造者の公開鍵 $Q$ で暗号化されており、製造者は自身の秘密鍵 $s$ で復号することができる。復号は $E_X$ の要素と秘密鍵 $s$ を用いて、式 (4) に示されるようにして平文 $T_x$ を求める。

$$T_x = (T_x + r_x Q) - s \cdot r_x G \tag{4}$$

この平文 $T_x$ は、製品の EPC と製造者 $M$・事業者 $X$ のブロックチェーンアドレスの排他的論理和を求め、それを楕円曲線 $E$ 上の点にしたものである。楕円曲線 $E$ 上の点にするときは、100 倍し、$i$ $(0 \leq i \leq 99)$ を加算していた。そのため、平文 $T_x$ を 100 で割り整数部分を求めることで、製品の EPC と製造者 $M$・事業者 $X$ のブロックチェーンアドレスの排他的論理和を求めたものが得られる。具体的には、式 (5) のようにして求められる。式 (5) 中の $A_M, A_X$ はそれぞれ製造者 $M$、事業者 $X$ のブロックチェーンアドレスを表したものである。

$$EPC \oplus A_M \oplus A_X = \left\lfloor \frac{T_x}{100} \right\rfloor \tag{5}$$

ここで、EPC は追跡する対象の製品を表すものである。また、製造者 $M$ のブロックチェーンアドレスは自身のブロックチェーンアドレスである。そのため、先に得られたものとこれらの排他的論理和をとることで、式 (6) に示されるように、事業者 $X$ のブロックチェーンアドレス $A_X$ を求めることができる。

$$A_X = (EPC \oplus A_M \oplus A_X) \oplus EPC \oplus A_M \tag{6}$$

$E_Y, E_Z$ についても同様の方法で、式 (7),(8) の形まで求めることができる。

$$EPC \oplus A_X \oplus A_Y = \left\lfloor \frac{M_y}{100} \right\rfloor \tag{7}$$

$$EPC \oplus A_Y \oplus A_Z = \left\lfloor \frac{M_z}{100} \right\rfloor \tag{8}$$

ここで、追跡対象の製品の EPC、式 (6) で求めた $A_X$ と式 (7) の左辺の排他的論理和をとることで、式 (9) に示されるように、事業者 $Y$ のブロックチェーンアドレス $A_Y$ を求めることができる。

$$A_Y = (EPC \oplus A_X \oplus A_Y) \oplus EPC \oplus A_X \tag{9}$$

式 (9) で $A_Y$ を求められたため、同様に式 (10) に示されるように、事業者 $Z$ のブロックチェーンアドレス $A_Z$ を求めることができる。

$$A_Z = (EPC \oplus A_Y \oplus A_Z) \oplus EPC \oplus A_Y \tag{10}$$

以上のようにして、製造者 $M$ は、製造者 $M$、事業者 $X$、事業者 $Y$、事業者 $Z$ の順で製品が流通したと判断できる。つまり、製造者 $M$ は、製品の流通経路を追跡することができる。

# 4　プライバシー保護と追跡性の動作実験

本章では、提案手法を利用することでプライバシー保護と追跡性の両立できていることを、シナリオを用いた動作実験を行うことで示す。

## 4.1　動作実験環境

提案手法の動作実験を行う環境を記す。プロトコルを実装する際に使用したスマートコントラクトの記述には Solidity [16] のバージョン 0.5.11 を使用した。動作実験は、Ethereum の統合開発環境である Remix [17] が提供する Ethereum のローカルネットワーク環境を利用して行う。製造者・事業者等のサプライチェーンの参加者は、直接的かつセキュアな通信手段が用意されていると仮定する。

本動作実験では、zk-SNARKs を用いたゼロ知識証明における証明鍵、検証鍵、証明情報を生成するために、ZoKrates [18] というツールを利用した。ZoKrates では、証明の生成スキームを文献 [19–21] で示されている方法のいずれかから選択することができる。本動作実験では、文献 [19] に基づいた証明生成スキームを選択する。zk-SNARKs では、楕円曲線のペアリング計算を行う必要がある。Ethereum では BN128 [22] と名前のつけられているペアリングパラメタに基づくペアリング計算を実行するための標準関数がスマートコントラクトを通じて提供されている。この標準関数を利用するためには、計算結果が表 4 に示す BN128 の群の要素数 $p$ を超えてはならない。そのため ZoKrates で使用する楕円曲線には、位数が $p$ となる式 (11) で定義されるものを利用する。

$$168700x^2 + y^2 = 1 + 168696x^2y^2 \ (mod\ p) \tag{11}$$

本動作実験で流通させる製品 $P$ とその製品に付与されている EPC 情報、および製品 $P$ の流通を行う製造者 $M$ および事業者 $X, Y, Z$ の情報を、それぞれ表 5 と表 6 に示す。これら製造者・事業者は、プロトコルに基づき正しく振る舞うものとする。また、製品 $P$ に付与されている EPC の事業者コードは、製造者 $M$ の事業者コードに一致しているとする。また、表 6 に記される製造者および事業者の他にも、偽造品の流通を試みる製造者 $M'$、流通に関与しない事業者 $S$ が登場する。

表 7 に製造者 $M$ の秘密鍵および公開鍵を記す。公開鍵は製品の所有者もしくは受領者として記録される暗号文を求めるために用いられる。秘密鍵は製品を追跡するために用いられる。

表 4: BN128 の位数 $p$

| 変数 | 数値 |
|---|---|
| $p$ | 21888242871839275222246405745257275088548364400416034343698204186575808495617 |

表 5: 動作実験シナリオにおける流通対象の製品とその EPC

| 流通対象 | EPC |
|---|---|
| 製品 $P$ | 045699511161791234567891 23 |

## 4.2 動作実験シナリオ

動作実験に利用するシナリオについて説明する。

### 4.2.1 シナリオ 1: 正常な製品の流通

このシナリオでは、偽造や複製等がされていない正常な製品を流通させる場合において、流通に参加する全ての事業者がプロトコルに基づき正しく振る舞うことで、正しく発送・受領の処理が実施できることを確認する。

具体的には、流通には製造者 $M$、事業者 $X, Y, Z$ が関与し、製品 $P$ を流通させる。流通は $M, X, Y, Z$ の順で行われる。シナリオを次に示す。また、シナリオの概略を表したものを図 12 に記す。

1. 製造者 $M$ が、製品 $P$ をブロックチェーンに登録する。
2. 製造者 $M$ が、次の受領者を事業者 $X$ に指定して製品 $P$ を発送する。
3. 事業者 $X$ が、自身が確かに事業者 $X$ であるという証明を提示することで、製品 $P$ を受領する。
4. 事業者 $X$ が、次の受領者を事業者 $Y$ に指定して製品 $P$ を発送する。
5. 事業者 $Y$ が、自身が確かに事業者 $Y$ であるという証明を提示することで、製品 $P$ を受領する。
6. 事業者 $Y$ が、次の受領者を事業者 $Z$ に指定して製品 $P$ を発送する。
7. 事業者 $Z$ が、自身が確かに事業者 $Z$ であるという証明を提示することで、製品 $P$ を受領する。

表 6: 動作実験シナリオに登場する事業者とそのアドレス

| 登場人物 | アドレス |
|---|---|
| 製造者 $M$ | 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C |
| 事業者 $X$ | 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB |
| 事業者 $Y$ | 0x583031D1113aD414F02576BD6afaBfb302140225 |
| 事業者 $Z$ | 0xdD870fA1b7C4700F2BD7f44238821C26f7392148 |

表 7: 製造者 $M$ の秘密鍵と公開鍵

| 鍵 | 数値 |
|---|---|
| 秘密鍵 | 0x9a893be169dd7a54255e7d845638cf2c1027d7a9f1f8ff26e6a8acc43760f56b |
| 公開鍵 | [“0xbbce34e251339fb5883efb22b66adef21b478266390f5822ccdc003953749fd”, “0x1bead44de06860038a9eb0ca221ac68a38357f91c4b0974119d32f8547f54244”] |

#### 4.2.2 シナリオ 2: 製造者の確認

このシナリオでは、任意の事業者が、その製品の製造者を確認できることを確認する。

具体的には、シナリオ 1 にて製品流通が終了した場合において、様々な事業者が製品 $P$ の製造者の情報の取得を試みる。シナリオを次に示す。また、シナリオの概略を表したものを図 13 に記す。

- 事業者 $X, Y, Z$ が製品 $P$ の製造者が $M$ であることを確認する。
- 流通に関与していない事業者 $S$ が製品 $P$ の製造者が $M$ であることを確認する。

#### 4.2.3 シナリオ 3: 偽造された製品の流通防止

このシナリオでは、ある事業者が、自身が製造者ではない製品をあたかも自身が製造者であるかのように装って流通させる場合において、その事業者が商品の流通を開始できないことを確認する。

具体的には、製造者 $M'$ が製品 $P$ を流通させることを試みる。シナリオを次に示す。また、シナリオの概略を表したものを図 14 に記す。

- 製造者 $M'$ が、製品 $P$ のブロックチェーンへの登録を試みる。しかし、エラーが発生し、製品をブロックチェーンに登録することはできない。

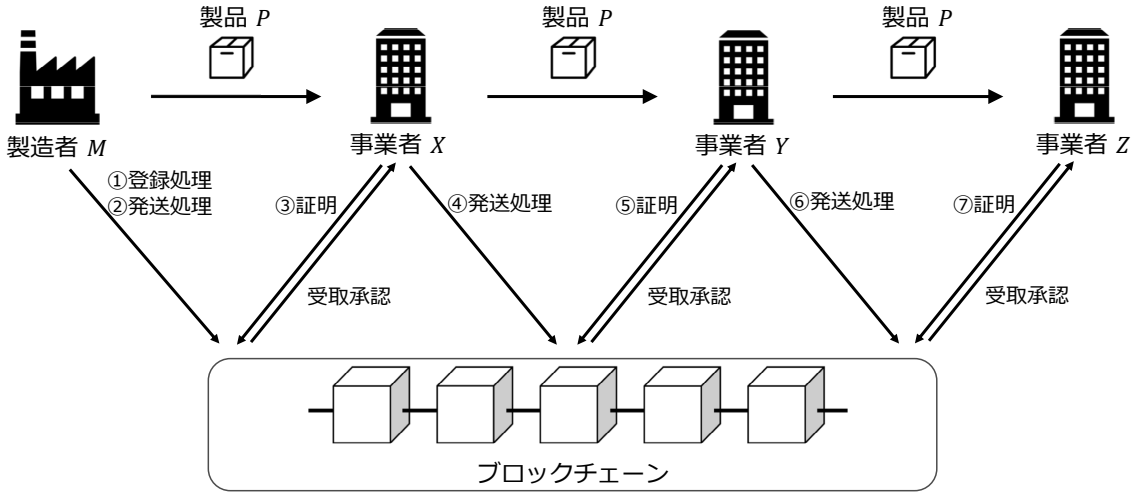


図 12: シナリオ 1 の概略図

#### 4.2.4 シナリオ 4: 流通情報のプライバシー保護

このシナリオでは、製品の流通に関与する事業者は、製造者と、製品の発送元である発送者および製品の発送先である受領者以外の一切の流通情報を知ることができないことを確認する。また、製品の流通に関与しない事業者は、製造者以外の一切の流通情報を知ることができないことを確認する。

具体的には、シナリオ 1 にて製品流通が終了した場合において、様々な事業者がブロックチェーンから流通情報の取得を試みる。シナリオを次に示す。

- 事業者 $X$ は、製造者 $M$ から受け取った事実、事業者 $Y$ に発送した事実はわかるが、事業者 $Y$・事業者 $Z$ 間の流通関係はわからない。
- 事業者 $Y$ は、事業者 $X$ から製品を受け取った事実、事業者 $Z$ に発送した事実はわかるが、製造者 $M$・事業者 $X$ 間の流通関係はわからない。
- 事業者 $Z$ は、事業者 $Y$ から製品を受け取った事実はわかるが、製造者 $M$・事業者 $X$ 間および事業者 $X$・事業者 $Y$ 間の流通関係はわからない。
- 流通に関与していない事業者 $S$ は、製造者 $M$・事業者 $X$ 間、および事業者 $X, Y, Z$ 間の流通関係はわからない。

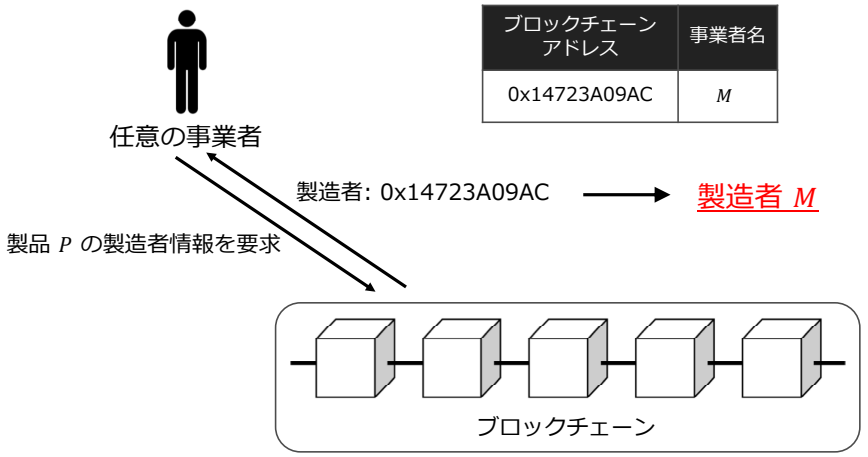


図 13: シナリオ 2 の概略図

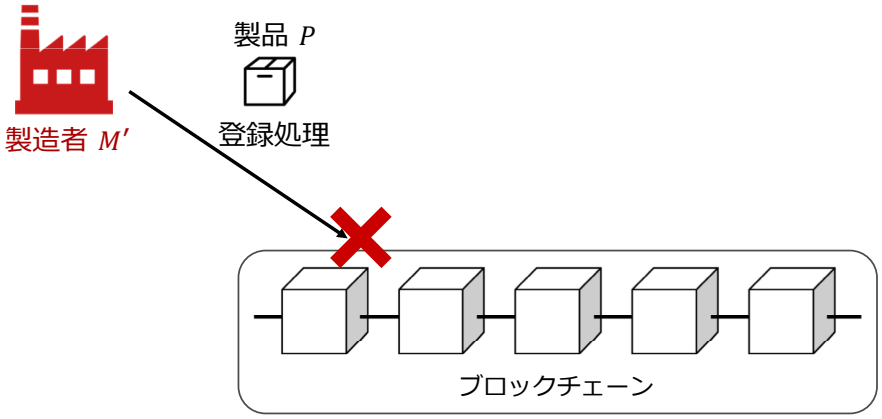


図 14: シナリオ 3 の概略図

#### 4.2.5 シナリオ 5: 製造者による流通経路の追跡

このシナリオでは、製造者によってその製品の全ての流通経路を特定できることを確認する。

具体的なシナリオは下記の通りである。また、シナリオの概略を表したものを図 15 に記す。

- 製造者 $M$ は、ブロックチェーンから暗号化された流通経路情報を取得する。
- 製造者 $M$ は、流通経路情報を復号し、それに適切な処理を施すことで、ブロックチェーンアドレスで表された流通経路を取得する。

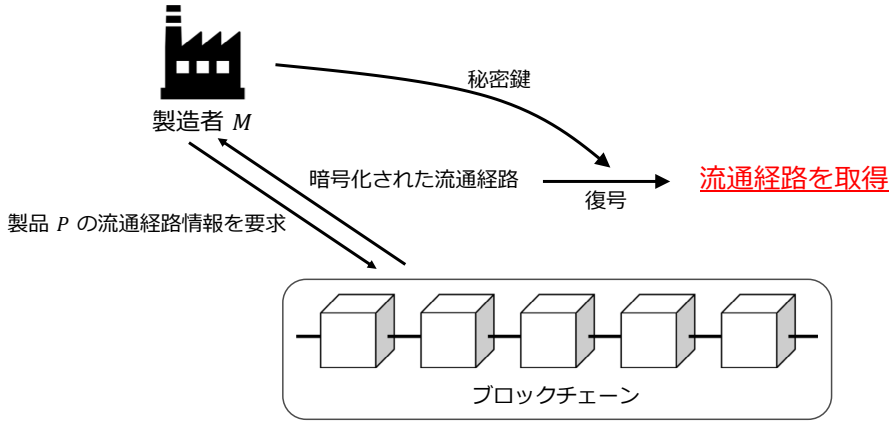


図 15: シナリオ 5 の概略図

## 4.3 動作実験結果

動作実験の結果を記す。Ethereum ネットワークから取得できるトランザクション情報を示しながら、4.2 節に記載のシナリオが想定通りに実行できることを述べる。また全てのシナリオの動作結果から、提案手法ではプライバシーの保護と追跡性が両立できていることを示す。

### 4.3.1 シナリオ 1: 正常な製品の流通

シナリオ 1 は、7 つの処理が実施される。各処理においてブロックチェーンに記録される情報を示すことで、シナリオ 1 が想定通りに実行されていることを示す。記録されている情報は、表 3 に示される変数を使って表す。

**製造者 $M$ が、製品 $P$ をブロックチェーンに登録する。**

製品 $P$ を表す EPC と製品 $P$ の製造者のアドレスがブロックチェーンに記録される。以降はこの EPC を通して、情報が記録される。

| | |
|---|---|
| EPC | 045699511161791234567891 23 |
| manufacturer | 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C |
| owner | null |
| recipient | null |
| status | owned |
| cntAddr | null |
| proof | null |

**製造者 $M$ が、次の受領者を事業者 $X$ に指定して製品 $P$ を発送する。**

事業者 $X$ を表す暗号文が受領者として記録される。また、事業者 $X$ が、受領および発送するための証明を検証するスマートコントラクトを示すアドレスも記録される。

| | |
|---|---|
| EPC | 045699511161791234567891 23 |
| manufacturer | 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C |
| owner | null |
| recipient | [[“0x1c9aeda8638112d8c76627faa0f803a5e1baab341472a7d2dae4f3eb4ae57fed”, “0x12cb0ca09845b7337dc061493c4446c3a984cb9a99d12dbcd77a04b1f37f0749”], [“0x26e2da36a0726f0d62323458fbcd0a38b5f75f88cb04d4eed5c33af18ffebd56”, “0x1dcded2f00a86fb5d6e1676cb0b2fc5ca1ba663b8d0498951b2691580262fa62”]] |
| status | shipped |
| cntAddr | 0x0FdF4894a3b7C5a101686829063BE52Ad45bcfb7 |
| proof | null |

変数 recipient に記録されている暗号文は、事業者 $X$ を表すものである。この暗号文は製造者 $M$ の公開鍵を用いて作成されている。つまり、この暗号文を復号するためには製造者 $M$ の秘密鍵が必要である。しかし、秘密鍵は他者に知られないように厳重に管理されている。そのため、製造者 $M$ 以外の者は、記録されている情報から、受領者のアドレスが事業者 $X$ のアドレス (0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB) であることを判断することは困難である。つまり、第三者からは受領者が事業者 $X$ であることはわからない。

**事業者 $X$ が、自身が確かに事業者 $X$ であるという証明を提示することで、製品 $P$ を受領する。**

事業者 $X$ は、自身を表す暗号文を計算し、変数 recipient に記録されている暗号文と比較することで、自身が受領者として指定されていることを確認する。そして、事業者 $X$ は変数 cntAddr に記録されているアドレス (0x0FdF4894a3b7C5a101686829063BE52Ad45bcfb7) が示すスマートコントラクトに、自身が確かに事業者 $X$ であるという証明を与える。このスマートコントラクトにより証明が検証され、正しい証明であると判断されると、事業者 $X$ を表す暗号文が所有者として変数 owner に記録される。また、与えられた証明も変数 proof に記録される。

| | |
|---|---|
| EPC | 045699511161791234567891 23 |
| manufacturer | 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C |
| owner | [[“0x1c9aeda8638112d8c76627faa0f803a5e1baab341472a7d2dae4f3eb4ae57fed”, “0x12cb0ca09845b7337dc061493c4446c3a984cb9a99d12dbcd77a04b1f37f0749”], [“0x26e2da36a0726f0d62323458fbcd0a38b5f75f88cb04d4eed5c33af18ffebd56”, “0x1dcded2f00a86fb5d6e1676cb0b2fc5ca1ba663b8d0498951b2691580262fa62”]] |
| recipient | [[“0x1c9aeda8638112d8c76627faa0f803a5e1baab341472a7d2dae4f3eb4ae57fed”, “0x12cb0ca09845b7337dc061493c4446c3a984cb9a99d12dbcd77a04b1f37f0749”], [“0x26e2da36a0726f0d62323458fbcd0a38b5f75f88cb04d4eed5c33af18ffebd56”, “0x1dcded2f00a86fb5d6e1676cb0b2fc5ca1ba663b8d0498951b2691580262fa62”]] |
| status | owned |
| cntAddr | 0x0FdF4894a3b7C5a101686829063BE52Ad45bcfb7 |
| proof | [“0x1d6d4b1ea2e5a8a787d09826ed43b99b41e33b56384ae73abb905265dcb66e85”, “0x098e061cf5b9fa761f10b24411841e2444913664e5a6d4747f545baae0aa09b5”], [[“0x26a6a4fffcd8abe3df47120bbd0d2046a3793266821e2f9c8bd4f114f78620cd”, “0x0fb400542399e0be20b436a3b1899ad063b2de90391d41e49c8b1b7428b1c6cf”], [“0x026c7a5258629941e85c7ac78cc8ac0041869852cc1c28ff6854ea60fb634981”, “0x26475001447c0fad9e75fbe3ee0681b9c2f2d6ab0547f4a0c01967b732217ca4”]], [“0x0d99fad8a303372386b957c5f2bd2075d9200f8dff29278fbf8b898851d57df3”, “0x20b5d618ed4d9d65150c7451b6d32f645b9ee8ad31d2048d39adc42787ff4d8e”] |

変数 proof に記録されている証明は、事業者 $X$ による自身が確かに事業者 $X$ であるという証明である。この証明は文献 [19] に基づいて生成されており、この証明には完全性、健全性、ゼロ知識性が担保されている。完全性が担保されていることから、証明に必要な知識を持っている証明者による証明は、検証結果が真となる。健全性が担保されていることから、証明に必要な知識を持たない者による証明は、検証結果が偽となる。また、ゼロ知識性が担保されていることから、この証明から証明の生成に用いられた数値を推測することは困難である。そのため、事業者 $X$ が真正な製品の受領者であることが保証され、かつ受領者が事業者 $X$ であることはわからない。

**事業者 $X$ が、次の受領者を事業者 $Y$ に指定して製品 $P$ を発送する。**

事業者 $X$ は、自身が確かに事業者 $X$ であるという証明を、変数 cntAddr に記録されているアドレス (0x0FdF4894a3b7C5a101686829063BE52Ad45bcfb7) が示すスマートコントラクトに与える。このとき、先に使用された証明を流用していないことを確認するために、変数 proof に記録されていた証明と与えられた証明が比較される。スマートコントラクトにより証明が検証され、正しい証明であると判断されると、事業者 $Y$ を表す暗号文が受領者として変数 recipient に記録される。また、事業者 $Y$ が、受領および発送するための証明を検証するスマートコントラクトを示すアドレスも記録される。

| | |
|---|---|
| EPC | 045699511161791234567891 23 |
| manufacturer | 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C |
| owner | [[“0x1c9aeda8638112d8c76627faa0f803a5e1baab341472a7d2dae4f3eb4ae57fed”, “0x12cb0ca09845b7337dc061493c4446c3a984cb9a99d12dbcd77a04b1f37f0749”], [“0x26e2da36a0726f0d62323458fbcd0a38b5f75f88cb04d4eed5c33af18ffebd56”, “0x1dcded2f00a86fb5d6e1676cb0b2fc5ca1ba663b8d0498951b2691580262fa62”]] |
| recipient | [[”0x26ded92e309bbc3865b6cf5c2d938e9fbc7c51ba2a5ea4c093556a8cc000ba07”, ”0xa7327e25596e7c679e2e2b994c8102b361f8e2bddae89451579f73d92cce88”], [”0x21375d2878a5bd88d970defcc7fd6145141830a53ddf03900b9d866b5f811d80”, ”0xaefd5a882579a5104ff8ca57291d7da10ae8d6b5b7e9a81b2d48b4f58e13fd8”]] |
| status | shipped |
| cntAddr | 0x8046085fb6806cAa9b19a4Cd7b3cd96374dD9573 |
| proof | [“0x1d6d4b1ea2e5a8a787d09826ed43b99b41e33b56384ae73abb905265dcb66e85”, “0x098e061cf5b9fa761f10b24411841e2444913664e5a6d4747f545baae0aa09b5”], [[“0x26a6a4fffcd8abe3df47120bbd0d2046a3793266821e2f9c8bd4f114f78620cd”, “0x0fb400542399e0be20b436a3b1899ad063b2de90391d41e49c8b1b7428b1c6cf”], [“0x026c7a5258629941e85c7ac78cc8ac0041869852cc1c28ff6854ea60fb634981”, “0x26475001447c0fad9e75fbe3ee0681b9c2f2d6ab0547f4a0c01967b732217ca4”]], [“0x0d99fad8a303372386b957c5f2bd2075d9200f8dff29278fbf8b898851d57df3”, “0x20b5d618ed4d9d65150c7451b6d32f645b9ee8ad31d2048d39adc42787ff4d8e”] |

変数 recipient に記録されている暗号文は事業者 $Y$ を表すものであり、事業者 $X$ を表す暗号文と同様に、第三者はこの暗号文から受領者が事業者 $Y$ であることはわからない。

また、発送する際に使用された証明は、事業者 $X$ による自身が確かに事業者 $X$ であるという証明である。事業者 $X$ が受領したときの証明と同様に、この証明により、事業者 $X$ が真正な製品の発送者であることが保証され、かつ発送者が事業者 $X$ であることはわからない。

**事業者 $Y$ が、自身が確かに事業者 $Y$ であるという証明を提示することで、製品 $P$ を受領する。**

事業者 $Y$ による製品 $P$ の受領は、事業者 $X$ による製品 $P$ の受領と同様の処理が行われる。そのため、この処理の詳細の記載は省略するが、動作実験結果から、事業者 $Y$ が受領できることを確認している。

また、これ以降も、事業者 $Y$ の発送処理、事業者 $Z$ の受領処理と続くが、いずれも事業者 $X$ の受領・発送処理と同様であるため、処理の詳細の記載は省略するが、動作実験結果から、事業者 $Y$ が発送できること、その後事業者 $Z$ が受領できることを確認している。

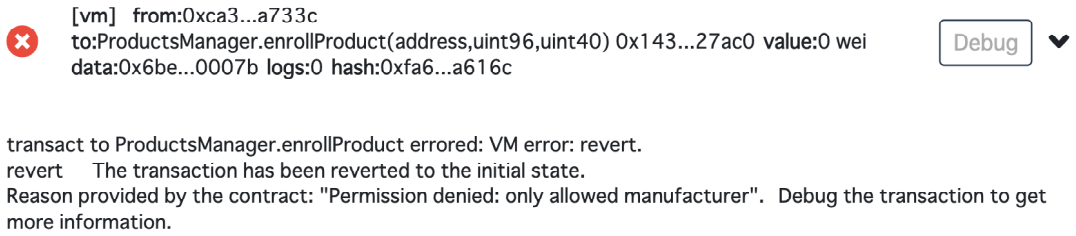


図 16: シナリオ 3 の動作実験結果

#### 4.3.2 シナリオ 2: 製造者の確認

ここでは、ブロックチェーンから取得した情報を示すことで、シナリオ 2 が想定通り実行できることを示す。

製品 $P$ の製造者を確認するために、スマートコントラクト ProductsManager から、製品 $P$ の EPC に対応づけられた製造者情報を取得する。これを、製品 $P$ の流通に関与した事業者 $X$,$Y$,$Z$ および流通に関与していない事業者 $S$ がそれぞれ実行する。この時、各事業者は同じ製造者情報を取得した。製造者情報を取得するための引数として使用した製品 $P$ の EPC と、取得した返り値を以下に示す。

| | |
|---|---|
| EPC | 045699511161791234567891 23 |
| 製造者情報 | 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C |

得られた返り値は、製造者 $M$ のアドレスに一致する。そのため、流通に関与する事業者、流通に関与していない事業者のいずれも、製品 $P$ の製造者が $M$ であることが確認できた。

#### 4.3.3 シナリオ 3: 偽造された製品の流通防止

シナリオ 3 では、製造者 $M'$ が製品 $P$ のブロックチェーンへの登録を試みる。このときに発生したエラーを示すことで、シナリオ 3 が想定通りに実行されていることを示す。図 16 が発生したエラーである。

図 16 に示されるエラーは、スマートコントラクト ProductsManager の関数 enrollProduct() を実行した時のものである。ここで示されているエラーメッセージは、 Permission denied: only allowed manufacturer. となっており、実行者である製造者 $M'$ が登録しようとした製品 $P$ の正規の製造者ではないために発生したエラーである。よって、自身が製造者ではない製品の流通は開始できない。

#### 4.3.4　シナリオ 4: 流通情報のプライバシー保護

ここでは、シナリオ 1 の後に、製品 $P$ の流通に関与した事業者 $X, Y, Z$ および流通に関与していない事業者 $S$ が、ブロックチェーンから情報を取得する。この取得した情報から、各事業者が特定可能な流通情報を示すことで、シナリオ 4 が想定通りに実行されていることを示す。

ブロックチェーンに記録されている情報は公開されているため、事業者 $X, Y, Z, S$ が取得できる情報は同じである。事業者 $X, Y, Z, S$ が取得できる情報は、4.3.1 節でブロックチェーンに記録された情報である。

**事業者 $X$ が特定できる流通情報**

事業者 $X$ がブロックチェーンに記録されている情報から特定できる流通情報を示す。

事業者 $X$ は、製造者 $M$ と共有した式 (2) の $r$ に該当する数値から作成した証明情報で、製品 $P$ を受領することができた。そのため、製品 $P$ を製造者 $M$ から受領したことを特定できる。また、事業者 $X$ は、事業者 $Y$ と式 (2) の $r$ に該当する数値を共有した後、ブロックチェーンに記録されている製品 $P$ の所有者が変更されたこと、つまり製品 $P$ が受領されたことを確認できる。そのため、事業者 $X$ は製品 $P$ を事業者 $Y$ に発送したことを特定できる。その後の事業者 $Y$・事業者 $Z$ 間の流通に関しては、所有者・受領者が暗号文で記録されていること、証明情報がゼロ知識性を有していることから、これらに使用された数値を推測することはできず、事業者 $Y$・事業者 $Z$ 間の流通関係は特定できない。

**事業者 $Y$ が特定できる流通情報**

事業者 $Y$ も、事業者 $X$ と同様に、製品 $P$ を事業者 $X$ から受領したこと、事業者 $Z$ に発送したことは特定できる。しかし、製造者 $M$・事業者 $X$ 間の流通関係は特定できない。

**事業者 $Z$ が特定できる流通情報**

事業者 $Z$ も、事業者 $X, Y$ と同様に、製品 $P$ を事業者 $Y$ から受領したことは特定できる。しかし、製造者 $M$・事業者 $X$ 間および事業者 $X$・事業者 $Y$ 間の流通関係は特定できない。

**事業者 $S$ が特定できる流通情報**

事業者 $S$ は、製品 $P$ の流通に関与しておらず、製品 $P$ の所有者・受領者が暗号文で記録されていること、証明情報がゼロ知識性を有していることから、これらに使用された数値を推測することはできない。そのため、事業者 $S$ は、製造者 $M$・事業者 $X$ 間、および事業者 $X, Y, Z$ 間の流通関係を特定することはできない。

#### 4.3.5　シナリオ 5: 製造者による流通経路の追跡

ここでは、シナリオ 1 の後に、2 つの処理を実施することで、最終的に流通経路の平文を取得する。それぞれの処理で取得できる情報を示すことでシナリオ 5 が想定通りに実行されていることを示す。

**製造者 $M$ は、ブロックチェーンから暗号化された流通経路情報を取得する。**

製品 $P$ の流通経路を確認するために、スマートコントラクト ProductsManager から、製品 $P$ の EPC に対応づけられた現時点までの所有者情報を取得する。所有者情報を取得するための引数として使用した製品 $P$ の EPC と、取得した返り値を以下に示す。

引数　045699511161791234567891 23

返り値 1　[[“0x1c9aeda8638112d8c76627faa0f803a5e1baab341472a7d2dae4f3eb4ae57fed”,
“0x12cb0ca09845b7337dc061493c4446c3a984cb9a99d12dbcd77a04b1f37f0749”],
[“0x26e2da36a0726f0d62323458fbcd0a38b5f75f88cb04d4eed5c33af18ffebd56”,
“0x1dcded2f00a86fb5d6e1676cb0b2fc5ca1ba663b8d0498951b2691580262fa62”]]

返り値 2　[[”0x26ded92e309bbc3865b6cf5c2d938e9fbc7c51ba2a5ea4c093556a8cc000ba07”,
”0xa7327e25596e7c679e2e2b994c8102b361f8e2bddae89451579f73d92cce88”],
[”0x21375d2878a5bd88d970defcc7fd6145141830a53ddf03900b9d866b5f811d80”,
”0xaefd5a882579a5104ff8ca57291d7da10ae8d6b5b7e9a81b2d48b4f58e13fd8”]]

返り値 3　[[“0x17713658eb31d82dd12adc0afc806c115b3a9c1ce94a281694fa83eece4d243”,
“0x327885ddf81c7bf0ffd92fb1403c6bb94755d7ae0cf007b046ab5f5d5f54271”],
[“0x1feb647abcf053481e573ed197ded6fb0afe5cc03a41c721de2ba7ebc3faf834”,
“0x2fc5e346be26fcd8c29989d44e044dd78112dea778a5811cc4669273518e605a”]]

取得した値はシナリオ 1 で事業者 $X, Y, Z$ を表す暗号文として記録されたものと等しい。返り値であるこれらの暗号文は、楕円曲線暗号の暗号文であり、この暗号文は楕円曲線上の点の組で表され、それぞれの点は $x, y$ 座標からなる。具体的には、返り値 1 は事業者 $X$ を表す暗号文であり、ここでは、返り値 1 の 1,2 行目を 1 つ目の点と呼び、3,4 行目を 2 つ目の点と呼ぶ。それぞれの点は $x, y$ 座標からなるため、返り値 1 の 1 行目が 1 つ目の点の $x$ 座標、2 行目が 1 つ目の点の $y$ 座標というようになる。このようにして、ブロックチェーンから暗号化された流通経路情報を取得できる。

**製造者 $M$ は、流通経路情報を復号し、それに適切な処理を施すことで、ブロックチェーンアドレスで表された流通経路を取得する。**

ここでは、まず、先に取得した暗号文を製造者の秘密鍵を用いて復号する。暗号文は楕円曲線暗号における暗号文であるため、以下で記される差や積を求める際の演算は楕円曲線上における演算である。

まず、返り値 1 の暗号文を復号する。暗号文は楕円曲線上の点の組で表されている。この点の組のうち、まず、1 つ目の点と製造者 $M$ の秘密鍵の積を求めると以下の楕円曲線上の点が得られる。

["0x25a6704a2a28828936cf00540a0b37fa2fc10d763453ab5664c6ad2b596abe4f",
"0xf491fb94c203028fadbe4d446faa0bae5180d77e3d2436253f4c0aa5d304ed5"]

次に、暗号文の 2 つ目の点と先の計算によって得られた積との差を求めると以下の楕円曲線上の点が得られる。

["0x254bebdca71f4545ac9cbdd8d7fa25b2d8723b98d2",
"0x26a2f52f05bb4e1f74746c2a835a7fe084d3310f16b06aaecc74db47501629c5"]

さらに得られた点の x 座標と製品 $P$ の EPC とこの暗号文で指定される所有者の 1 つ前の所有者のアドレス、つまり製造者のアドレスの排他的論理和を求めると以下が得られる。

0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db

これは事業者 $X$ のアドレスである。このことから、この流通における受領者は事業者 $X$ であることがわかる。これと同様の操作を繰り返すことで、以下のアドレスのリストが得られる。

| index | address |
|---|---|
| 1 | 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db |
| 2 | 0x583031d1113ad414f02576bd6afabfb302140225 |
| 3 | 0xdd870fa1b7c4700f2bd7f44238821c26f7392148 |

これらは事業者 $X, Y, Z$ のアドレスである。以上から製品 $P$ の流通経路を追跡できることが確認できた。

# 5　おわりに

サプライチェーンのグローバル化が進んでおり、これに伴う問題として、サプライチェーンにおける追跡性の低下が懸念されている。このような状況下において、追跡性の向上を目的として、ブロックチェーンを用いた製品情報を管理する手法が提案されている。しかし、ブロックチェーンに記録されている情報は誰もが閲覧可能であり、流通情報が公開されてしまうといったプライバシーに関わる問題が存在する。そこで、本報告では、流通情報を秘匿にしつつも、真正な事業者の間で製品を流通可能な方法を提案した。具体的には、既存手法では、所有者情報としてブロックチェーンのアドレスを記録することで、流通情報を管理していたため、この記録されるブロックチェーンのアドレスを暗号化を用いて隠蔽した。また、本報告で提案した手法では、発送者・受領者の間で値を共有し、その値を知っている者を真正な事業者であるとして、製品の受領・発送を許可する。この値を知っていることにゼロ知識証明を用いることで、真正な事業者であることを確認した。また、シナリオに基づく動作実験の結果、流通している製品が、真正な製造者が製造したものであること、真正な事業者のみで流通されていることが保証されること、問題発生時には製造者が所在を特定できること、を確認した。

今後の課題としては、提案手法の定量的な評価として、流通の際に必要な Ethereum の手数料を計測し、既存手法である POMS と比較することが挙げられる。評価を通じて、提案手法を用いることで、POMS に対してわずかに手数料が増加するだけで、プライバシーの保護ができることを示したい。また、本報告で使用したブロックチェーンチェーンプラットフォームである Ethereum は、仕様上、トランザクションの実行者のアドレスが公開情報なってしまう。現在の提案手法は、発送者、受領者がそれぞれ自らトランザクションを実行する実装になっているため、アドレス情報が公開情報となってしまう。そこで、発送者・受領者と、トランザクション実行者を分けるなどして、アドレス情報が公開されないようなトランザクション処理の仕組みを検討する予定である。

# 謝辞

本報告を終えるにあたり、日頃よりご指導、ご教授くださいました大阪大学大学院情報科学研究科の村田正幸教授に心より感謝申し上げます。ならびに、本報告の作成にあたり、ご多忙にもかかわらず、丁寧なご指導をしてくださいました四條能伸氏に深く感謝申し上げます。加えて、大阪大学大学院情報科学研究科の矢内直人助教には、セキュリティに関するご助言をいただきましたこと、誠に感謝申し上げます。また、大阪大学大学院情報科学研究科の荒川伸一准教授、大阪大学大学院情報科学研究科の大下裕一准教授、大阪大学大学院経済学研究科の小南大智助教、大阪大学大学院情報科学研究科の大歳達也特任教授には、平素より適切な助言をしていただきましたこと、心より感謝いたします。最後に、日頃から様々な面でご助言・ご助力くださいました志垣沙衣子氏、山内雅明氏をはじめとする村田研究室の皆様に感謝いたします。

# 参考文献


[1] OECD/EUIPO, *Trade in Counterfeit and Pirated Goods*. Illicit Trade, Paris: European Union Intellectual Property Office, 2019.

[2] Centers for Disease Control and Prevention, "Multistate Outbreaks of Shiga toxin-producing Escherichia coli O26 Infections Linked to Chipotle Mexican Grill Restaurants (Final Update)." `https://www.cdc.gov/ecoli/2015/o26-11-15/index.html`, Feb. 2016. [Online; accessed 10-February-2020].

[3] S. Li and B. Lin, "Accessing information sharing and information quality in supply chain management," *Decision Support Systems*, vol. 42, pp. 1641–1656, Dec. 2006.

[4] K. Toyoda, P. Takis Mathiopoulos, I. Sasase, and T. Ohtsuki, "A Novel Blockchain-Based Product Ownership Management System (POMS) for Anti-Counterfeits in the Post Supply Chain," *IEEE Access*, vol. 5, pp. 17465–17477, 2017.

[5] H. M. Kim and M. Laskowski, "Toward an ontology-driven blockchain design for supply-chain provenance," *Intelligent Systems in Accounting, Finance and Management*, vol. 25, pp. 18–27, Mar. 2018.

[6] "modum." `https://modum.io/`. [Online; accessed 10-February-2020].

[7] "Chronicled." `https://www.chronicled.com/`. [Online; accessed 10-February-2020].

[8] "IBM Blockchain for supply chain." `https://www.ibm.com/blockchain/industries/supply-chain`. [Online; accessed 31-January-2019].

[9] N. Kshetri, "1 Blockchain's roles in meeting key supply chain management objectives," *International Journal of Information Management*, vol. 39, pp. 80–89, 2018.

[10] Feng Tian, "An agri-food supply chain traceability system for china based on rfid blockchain technology," in *Proc. 2016 13th International Conference on Service Systems and Service Management (ICSSSM)*, pp. 1–6, June 2016.

[11] H. Huang, X. Zhou, and J. Liu, "Food supply chain traceability scheme based on blockchain and epc technology," in *Proceedings of Smart Blockchain*, pp. 32–42, 2019.

[12] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System." `https://bitcoin.org/bitcoin.pdf`, 2008. [Online; accessed 10-February-2020].

[13] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger." `https://gavwood.com/paper.pdf`, 2014. [Online; accessed 10-February-2020].

[14] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *Proceedings of 23rd USENIX Security Symposium (USENIX Security 14)*, pp. 781–796, Aug. 2014.

[15] L. Washington, *Elliptic Curves*. Chapman and Hall/CRC, 2008.

[16] "Solidity." `https://solidity.readthedocs.io/`. [Online; accessed 10-February-2020].

[17] "Remix - Ethereum IDE." `https://remix.ethereum.org`. [Online; accessed 10-February-2020].

[18] "ZoKrates." `https://github.com/Zokrates/ZoKrates`. [Online; accessed 10-February-2020].

[19] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly Practical Verifiable Computation," in *Proceedings of 2013 IEEE Symposium on Security and Privacy*, pp. 238–252, May 2013.

[20] J. Groth, "On the Size of Pairing-Based Non-interactive Arguments," in *Proceedings of Advances in Cryptology – EUROCRYPT 2016*, pp. 305–326, Apr. 2016.

[21] J. Groth and M. Maller, "Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs," in *Proceedings of Advances in Cryptology – CRYPTO 2017*, pp. 581–612, July 2017.

[22] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Proceedings of Selected Areas in Cryptography*, pp. 319–331, 2006.