Received February 9, 2017, accepted March 31, 2017, date of publication June 28, 2017, date of current version September 19, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2720760

A Novel Blockchain-Based Product Ownership Management System (POMS) for Anti-Counterfeits in the Post Supply Chain

KENTAROH TOYODA[1], (Member, IEEE), P. TAKIS MATHIOPOULOS[2], (Senior Member, IEEE), IWAO SASASE[1], (Senior Member, IEEE), AND TOMOAKI OHTSUKI[1], (Senior Member, IEEE)
[1]Keio University, Kanagawa 223-8522, Japan
[2]National and Kapodestrian University of Athens, 15784 Athens, Greece

Corresponding author: Kentaroh Toyoda (toyoda@ohtsuki.ics.keio.ac.jp)

This work was supported by the Kakenhi from the Ministry of Education, Sport, Science and Technology, Japan, under Grant 16H07168.

ABSTRACT For more than a decade now, radio frequency identification (RFID) technology has been quite effective in providing anti-counterfeits measures in the supply chain. However, the genuineness of RFID tags cannot be guaranteed in the post supply chain, since these tags can be rather easily cloned in the public space. In this paper, we propose a novel product ownership management system (POMS) of RFID-attached products for anti-counterfeits that can be used in the post supply chain. For this purpose, we leverage the idea of `Bitcoin`'s blockchain that anyone can check the proof of possession of balance. With the proposed POMS, a customer can reject the purchase of counterfeits even with genuine RFID tag information, if the seller does not possess their ownership. We have implemented a proof-of-concept experimental system employing a blockchain-based decentralized application platform, `Ethereum`, and evaluated its cost performance. Results have shown that, typically, the cost of managing the ownership of a product with up to six transfers is less than U.S. $1.

INDEX TERMS Anti-counterfeits technology, POMS (products ownership management system), blockchain, Ethereum, security.

I. INTRODUCTION

Counterfeiting products, such as branded goods, is one of the most important and difficult issues to deal with in national/international markets. This has been recognized for more than a decade now, as the OECD (Organisation for Economic Co-operation and Development) announced in 2007 that the counterfeits in international trade could amount to about US$250 billion [1]. Because of the rapid development of e-and i-commerce, clearly there exists an urgent demand to develop anti-counterfeits systems. On the other hand, for over a decade, RFID (Radio Frequency IDentification) as a key technology in the IoT (Internet of Things) world, has received a lot of attention since it can be used to detect counterfeits which are inserted into the supply chain, e.g. [2]–[5]. In the RFID-enabled supply chain, an EPC (Electronic Product Code) is assigned to each product and is written into an RFID tag. Such tag-attached products are shipped from manufacturers to the supply chain parties. During the transportation process, each involved party interrogates RFID tags and adds extra evidence data into tags. In this way, the next party can check whether or not the products have passed through the legitimate supply chain. If any inconsistency is found, such products may be considered as counterfeits.

However, once the products reach the end of the supply chain and are displayed in retail stores, their genuineness is no longer guaranteed, as anyone who has an RFID reader can interrogate and clone tags' information. Therefore, it is important to develop anti-counterfeits systems that work even when the tag's information is cloned in the post supply chain.

In this paper, we propose a novel product ownership management system (POMS) which makes the efforts of counterfeiters to clone genuine tags redundant since they cannot prove the possession of products on this system. For this purpose, we leverage the idea of `Bitcoin`, a decentralized cryptocurrency system in which the possession of user's balance can be proven in the public ledger referred to as *blockchain* [6]. In particular, by borrowing the ideas first presented in the "proof of possession of balance" used in

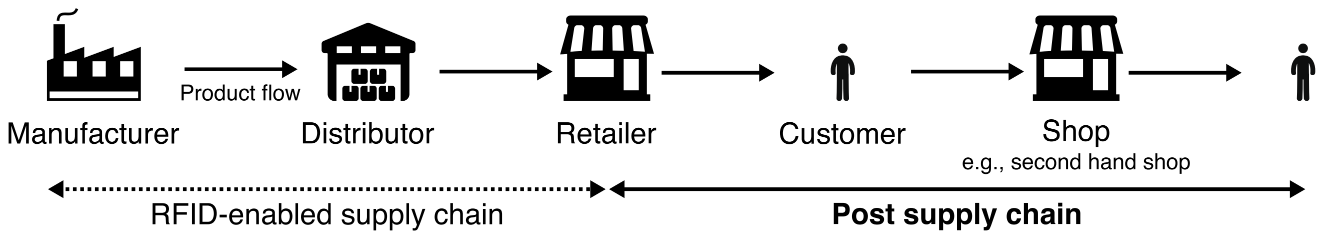


FIGURE 1. Structure of the RFID-enabled and post supply chains.

Bitcoin, we introduce here the concept of "proof of possession of products". Furthermore, several issues must first be identified and then be addressed for the successful realization of POMS with blockchain technology. For example, only the legitimate manufacturers can claim the first ownership of their own products. To comply with such requirements, in this paper we propose a novel blockchain-based POMS for anti-counterfeits which works very effectively for the post supply chain. Firstly, the overall practical system requirements are identified. Then, we introduce a full-fledged protocol that enables each party, including supply chain partners and customers, to transfer and prove the ownership of RFID tag-attached products. An important advantage of the proposed POMS is that customers can reject the purchase of counterfeits, even with a genuine EPC, under the condition that the seller does not possess their ownership. Based on the proposed protocol, a proof-of-concept experimental system has been implemented on the Ethereum platform. Performance evaluation results have shown that the cost for managing products with the proposed POMS is less than US$1 when the number of owner transfers is not more than six.

The rest of this paper is structured as follows. Section II presents the system model, reviews the previously published literature and deals with the basic issues of how blockchain and Bitcoin technologies can be used in the context of our research. The actual operation of the proposed POMS is described with details in Section III. Performance evaluation can be found in Section IV, whereas open problems regarding our POMS are discussed in Section V. Finally, the conclusions are presented in Section VI.

II. PRELIMINARIES

In this section, firstly, the system model is introduced. Then, an overview of the previously published papers and the motivation behind our work will be presented. Thirdly, the fundamentals of blockchain and Bitcoin technologies, and how these can be considered within the blockchain operation are explained.

A. SYSTEM MODEL

Fig. 1 illustrates the typical product flow consisting of two chains, namely the RFID-enabled and post supply chains.[1] The first one is typically composed of three parties, i.e., manufacturers, distributors, and retailers [7]. A manufacturer creates, composes, and ships products to the distributors while they decompose the received products and ship them further to the retailers. In the post supply chain, retailers stock and sell their products to customers who in turn may resell them, e.g. at a second hand shop or over the Internet.

We consider that each supply chain party has EPCglobal Class 1 Gen 2 (C1G2) compatible RFID readers[2] A manufacturer assigns an EPC to every product and EPCs are written into tags attached to the products so that any party can recognize products when arrived. It is also assumed that every party has an Internet connection via computers and/or smartphones/tablets.

B. PREVIOUS WORK AND MOTIVATION

For over a decade now, RFID technology has been integrated into the supply chain for anti-counterfeits. The first systematic RFID-based approach for anti-counterfeits in food and drug industry was proposed by FDA (Food and Drug Administration) [9]. In their proposal, each supply chain party is equipped with RFID readers and keeps track of shipping and receiving events for each product. In this way, the supply chain parties have the ability to track and trace the product flow of products. However, as it was pointed out in [2], such an approach is vulnerable against *cloned tags*. Specifically, once RFID tags attached to the genuine products are copied by an attacker, counterfeits with cloned RFID tags can be inserted in the supply chains. In this way, counterfeits with cloned tags cannot be identified by the aforementioned track and trace approach. So far, the research efforts dealing with this problem can be classified into two categories. The first one involves the development of secure tag distribution schemes so that an attacker cannot copy the contents of tags in the supply chains, e.g. see [7], [10]–[14]. Among these works, perhaps the most important one is a secret sharing based key distribution scheme [7]. In this scheme, the tag's information is encrypted with a symmetric encryption scheme and an encryption key is split into multiple shares by a secret sharing scheme [7]. The secret sharing scheme realizes that one can extract the key if he/she can obtain more than certain amount of unique shares [15]. An encrypted EPC and the

[1] In this paper, we assume that in the considered post supply chain and for the vast majority of products, any party involved is allowed to resell them. However, it is noted that there exist some categories of products, such as prescription drugs, for which it is illegal to resell them through secondary channels, e.g. the Internet.

[2] http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2, whereas each post supply chain party has a C1G2 RFID, QR (Quick Response) code [8], or NFC (Near Field Communication) tag reader.

shares of a key is written into a tag on the product. After receiving products, an authorized partner interrogates tags and recovers the key from sufficient number of shares. Then, the EPCs are decrypted with the extracted key. The second category deals with cloned tags detection schemes in the RFID-enabled supply chains, e.g. see [2]–[5], [16]–[18]. For example, in [4], when a tag arrives at a supply chain party, it writes an arrival evidence to the next available position in the tag memory. Then, the party requests a detector who can obtain information of a supply chain to check whether or not the information in the tag is valid. By doing so, a detector can identify counterfeits if any inconsistency is found, i.e., written positions are wrong and/or written words are not matched.

However, once the products are for sale in retail stores, i.e., in the post supply chain scenario, it is not feasible to execute any such track and trace methods nor secure EPC distribution schemes. In this case, none of the currently employed track and trace methods can guarantee the tag-attached product's genuineness because they leverage the tag's secret information. In addition, it is obvious that once they are displayed in public, any secure EPC distribution does not make much sense. Hence, EPCs are not assured any more to be unique and genuine. Thus, it is crucial to consider the counterfeits detection even in the post supply chain. For this, an anti-counterfeits scheme that a customer can check the legitimacy of products was proposed in [19]. However, this approach requires a special computational tag for each product and thus publicly available off-the-shelf tags cannot be used. Moreover, if a customer sells such products to a second hand shop or at the auction, it will also require customers to register their certificates at PKI (Public Key Infrastructure), which is not realistic.

Motivated by the above, our basic idea here is to introduce here the concept of proving ''the possession of products'' and designing a novel POMS which manages and tracks the possession of products starting from their manufacturers to the current owners. With such a scheme, counterfeits may be detected if a party cannot prove the possession of claimed products. For example, let's assume that a customer wants to buy a branded product through a second hand shop. A genuine EPC is attached to this selling product but actually , in reality, it is a counterfeit. In this case, it is impossible for the customers to check whether or not the seller possesses the ownership of the claimed EPC. To check this, we need to verify if (i) The initial owner of the product with the claimed EPC must be its manufacturer, and (ii) The current owner of the product with the claimed EPC must be the seller. The obvious and straight-forward way to validate the two conditions is that manufacturers construct a large-scale system which manages the ownership of their products. However, this is a very complex and costly procedure since it must also be secured against attacks, e.g. DDoS (Distributed Denial of Service) attacks [20]. Furthermore, such system typically requires the involvement of many parties and rather complex procedures, including the consumers to register their identification, that is, dealing with sensitive data, e.g. ID/password pairs. Therefore, a more efficiently operating POMS is required for dealing with such issues and at the same time being compatible with the RFID-enabled supply chain.

C. BLOCKCHAIN TECHNOLOGY

To realize such a POMS, we leverage the idea of `Bitcoin`, a decentralized cryptocurrency system in which the possession of user's balance can be proven in the public ledger referred to as *blockchain* [6]. Specifically, we replace its concept of ''proof of possession of balance'' with an equivalent concept which we will refer to as ''proof of possession of products''.

A few startup companies have just started (or soon plan to start) services with the blockchain for managing the genuineness of products [21], [22]. For example, `Everledger` appears to be the most successful service specific for diamonds [21]. `Everledger` offers a permanent ledger for diamond certification and related transaction history. This special ledger is used for verification for insurance company, owners, claimants and law enforcement. Another example is `Blockverify` which appears to use the blockchain technology for pharmaceuticals and luxury items[3] [22]. However, no details whatsoever are available publicly about the protocol of such commercial services, meaning that its security and privacy issues cannot be reviewed in the context of our paper. Nevertheless, it is clearly of importance to propose a rigorous and transparent protocol for blockchain-based POMS. In order to do so, the overview of the blockchain technology will be presented next and then our POMS will be described in detail in the next section.

1) BITCOIN AND BLOCKCHAIN

We first focus on `Bitcoin`, a decentralized cryptocurrency where any trusted authority, e.g. a bank, does not exist, because `Bitcoin` is the first successful system using blockchain technology. With the blockchain, `Bitcoin` allows the users to prove the ownership of their balance without any authentication and a centralized authority. It is useful first to briefly explain how a user transfers Bitcoin to another user. In `Bitcoin`, a data structure referred to as *transaction* is used to identify the amount to be transferred, sender(s), and recipient(s). Fig. 2 illustrates an example of such `Bitcoin` transactions, which presents the detailed procedure of user A transferring his/her Bitcoin to user B and then user B transferring his/her Bitcoin to user C, as indicated in the first transaction in Fig. 2. For user A to transfer his/her Bitcoin to user B, user A's address calculated from A's public key, the source of Bitcoin (unspent balance user A owns), and its signature calculated by user A's private key are specified in the inputs part of a transaction. At its outputs part, user A specifies the amount to be transferred and user B's address which is calculated from user B's public key. User A also specifies his/her own address or possibly his/her newly created address owned by user A to receive

[3]To the best of our knowledge, the details of `Blockverify` including its possible operation, are not available publicly.

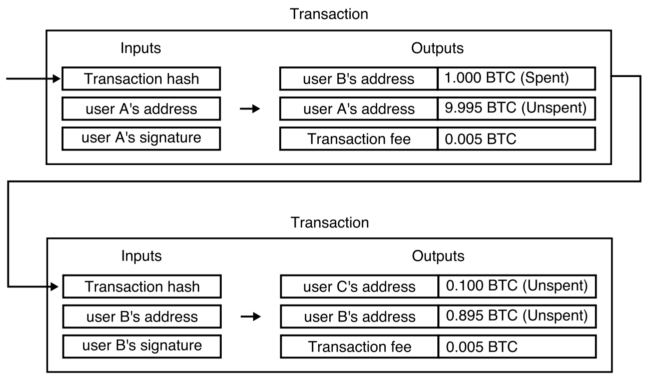


FIGURE 2. An example of Bitcoin transactions between users *A* and *B* as well as between *B* and *C*.

the change. As illustrated in the second transaction in Fig. 2, when user *B* transfers Bitcoin to user *C*, user *B* follows the same procedure that user *A* did. It is well-known that in `Bitcoin`, any user can check every transaction. Hence, if the same address is used for several transactions, this could be a problem with the privacy issue. To reduce the effects of this privacy issue, it is recommended for users to generate a fresh address for every transaction. Since it is a complex task to manage multiple addresses, an application called *wallet* is used to efficiently store and generate public/private key pairs for every transaction.

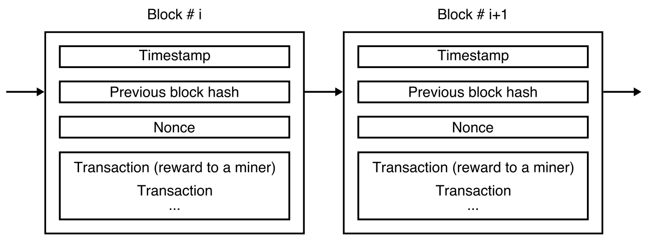


FIGURE 3. A sequence of blocks used to construct a blockchain.

Since `Bitcoin` is decentralized, a consensus algorithm is necessary so that every user agrees on the order of transactions. Without a consensus, any malicious user might try to use already spent balance twice, which is so-called double-spend. `Bitcoin` achieves (probabilistic) consensus by introducing the *blockchain* concept, which is an ever-growing chain of blocks that contain approved transactions. Fig. 3 illustrates a sequence of blocks used to construct a blockchain. A block is created by imposing potential block creators, referred to as *miners*, to solve the computational puzzle that is difficult to solve but is easy to check. More specifically, miners are required to find a nonce (number used once) that satisfies its (SHA-256) hash value together with a reference to the previous block and a set of the unapproved transactions lowers the target value. Since the previous block is required to generate the next block, this creates an ever-growing chain of blocks, i.e., blockchain. In order for the miners to follow such protocol, the following incentive mechanism is adopted: The first solver of the block acquires the newly minted `Bitcoin` as a reward, and in addition, all transaction fees which are included in the block.

Although such a blockchain-based decentralized consensus mechanism is firstly used for "currency", it can be extended to other applications that need to be *publicly verifiable*. Clearly, POMS is such an application as it can be realized based on the blockchain by substituting the concept of "currency" with that of a "product". This notion has been already embodied as a CC (Colored Coin) protocol in that any `Bitcoin` user can issue its own currency. For example, a CC may represent an asset or a property, e.g. a car or a house. Most of CC protocols, such as [23] and [24], were built by leveraging `Bitcoin`'s scripting function, in which a sender can specify rules to send a Bitcoin. In another approach, it was suggested to construct a container tracking system with a CC protocol by issuing and exchanging "I have a container" and "I received a container", tokens originally issued by manufacturers and other parties [25]. However, it should be emphasized that the CC protocol cannot be used for the POMS because it lacks the operating requirements necessary for its successful implementation. Although these requirements will be presented with details in Section III-A, it is worthwhile mentioning here that for POMS it is necessary to provide incentives to the cooperative parties, so that the parties follow the protocol. Unfortunately, it is not possible to fulfill such requirement because of its rather simple CC protocol available on `Bitcoin`. Therefore, to implement a well functioning POMS, we must choose another blockchain-based consensus platform which can support the execution of any code. To date, `Ethereum` is the only platform that fulfills this condition [26], and thus its operation will be described next.

2) ETHEREUM

`Ethereum` is a blockchain-based decentralized cryptocurrency where any code execution is possible [26]. To enable code execution, the memory storage is required on accounts. For this reason, `Ethereum` has two types of accounts: CA (Contract Account) and EOA (Externally Owned Account). On the one hand, an EOA manages its own balance. On the other hand, a CA has its own code as well as storage and executes the code when a message is received from another CA or EOA. To write a code for a CA, a scripting language called `Solidity` is typically used [27]. The code written in `Solidity` is compiled into the stack-based programming language so that a CA can execute. Since any algorithm can be described on `Ethereum` with `Solidity`, the POMS for anti-counterfeits can be also implemented on `Ethereum`.

Supporting the programming language means that an attacker can make miners keep computing meaningless codes, e.g. causing an infinite loop, aiming for wasting computing resources and energy. To avoid this attack, any code execution that changes the storage requires a sender contract sufficient amount of "gas" according to procedures, e.g. data amount and the number of computational steps [28]. If "gas"

TABLE 1. An example of SGTIN-96 (serialized global trade item number), where the total length of company prefix and item reference is 44 bits.

Header (8 bits)	Filter Value (3 bits)	Partition (3 bits)	Company Prefix (20-40 bits)	Item Reference (24-4 bits)	Serial Number (38 bits)
00110000	001	101	00· · · 10	10· · · 01	00· · · 01

runs out during the code execution, the state will be reverted to the original one, but the cost for "gas" is not returned to the sender.

III. THE PROPOSED POMS

In this section, an overview of the proposed POMS will be presented. In particular, the key system requirements will be first outlined followed by the pseudo-codes of the implemented contracts in `Ethereum`. Next, the details of the algorithmic procedures between all parties necessary for the realization of the proposed POMS will be identified. As it will become apparent in the last part of this section, the main advantage of the proposed POMS is that it makes tags cloning meaningless since even if tags are cloned by the counterfeiters, they cannot prove the ownership of the products.

A. POMS OPERATIONAL REQUIREMENTS

Before describing the POMS protocols in detail, we present the key requirements and explain their necessity for the proper operation of the proposed product ownership proof system.

1) Only the legitimate manufacturers are able to claim the initial ownership (origin) of products (EPCs);
2) Each manufacturer can declare only their own products;
3) The events "Shipped" and "Received" can be separated;
4) A manufacturer must give some incentive to each party who follows the POMS protocol.

The first requirement is necessary to avoid any non-authorized parties including counterfeiters from illegally issuing the ownership of products.

The second one is also required, since without it, the following case could be possible: A manufacturer, M_1, claims the initial ownership of manufacturer, M_2's, products. This can be avoided by leveraging the company prefix specified in EPC format of SGTIN-96, which is the 96 bits EPC format to identify products. Table 1 shows the message format of SGTIN-96. As it can be seen from this table, SGTIN-96 includes the company prefix which identifies its manufacturer. In reality, each manufacturer may enroll its own company prefix in `GS1` in the RFID-enabled supply chain. By leveraging this, manufactures can claim the initial ownership of only their own EPCs whose company prefix is enrolled by `GS1`.

Regarding the third requirement, the reason why we divide the ownership transfer process is to avoid the situation where the current owner sends the product to the recipient while it does not arrive at its destination. In this case, if only a function that simply transfers the ownership of a product were implemented, the following undesired case might occur: The ownership is transferred, however, the product itself is not arrived at the recipient.

The last requirement will enable the proper operation of the overall system, because every party except for manufacturers has to pay fees to issue a contract which transfers the ownership of products. Hence, without incentive mechanisms, the non-manufacturer parties will make a loss and eventually have no motivation to follow the POMS protocol.

B. IMPLEMENTED CONTRACTS

To satisfy the above requirements, we have implemented two contracts, namely `MM` (`ManufacturersManager`) and `PM` (`ProductsManager`). On the one hand, `MM` offers functions for managing the information of manufacturers, e.g. enrollment of a company prefix registered in `GS1` and manufacturer's address. On the other hand, `PM` is operated by each manufacturer and offers functions to manage the information of products, e.g. enrollment of a new product and ownership transfer.

In contrast to `PM`, in `MM` we assume the existence of an administrator who manages the manufacturers' information. To avoid impersonation, only the administrators can modify any manufacturer's information. One of the administrative candidates is `GS1`, because it manages company prefix. Although this may break the assumption of fully decentralized system, it is inevitable in order to avoid impersonators, e.g. counterfeiters, from registering themselves as if they are legitimate manufacturers. Actually, it might be possible to make our `MM` decentralized by leveraging the notion of `Namecoin` [29]. `Namecoin` is a decentralized domain name system and avoids "massive" registration by imposing cost for enrollment. However, if our `MM` might be constructed by introducing registering fee like `Namecoin`, there could be a chance for counterfeiters to illegally register themselves as genuine companies by paying the appropriate fees. However, it still might be possible to make `MM` decentralized. This is one of the open questions regarding the blockchain-based POMS.

Next, the pseudo-codes of `MM` and `PM` will be presented with details. The key shortened codes written in `Solidity` can be found in Appendix.

1) MANUFACTURERS MANAGER (`MM`)

`MM` is mainly composed of two functions: (i) enrolling the manufacturer's information in the blockchain and (ii) checking the authorship for a requesting manufacturer to enroll the product's EPC.

Algorithm 1 Pseudo-Code of `enrollManufacturer()` Enrolling a Manufacturer's Information on a Blockchain

```
1: Inputs:
   Manufacturer $M$'s address ($A_M$), company prefix (companyPrefix), name (companyName), and valid duration (validDuration)
2: if the sender of this transaction is an Admin (like GS1) then
3:     Enroll manufacturer $M$'s information, i.e., $A_M$, companyPrefix, companyName, and validDurationon the blockchain
4: else
5:     Do nothing
6: end if
```

Alg. 1 shows the pseudo-code of `enrollManufacturer()`, which enrolls the manufacturer's information required when its product is stored in the blockchain. Since our POMS requires that only one administrator, e.g. `GS1`, can enroll the manufacturer's information, this condition is checked at step 2. If it is `True`, then the admin enrolls the manufacturer's information in the blockchain.

Algorithm 2 Pseudo-Code of `checkAuthorship()`, Which Checks Whether or Not a Message Sender Possesses the Authorship of a Claimed EPC

```
1: Inputs: The message sender's address ($A_{\text{msg}}$) and EPC
2: Output: A boolean (True or False)
3: companyPrefix$_{\text{msg}}$ ← Get the message sender's companyPrefix through the blockchain
4: if companyPrefix described in the claimed EPC is the same with companyPrefix$_{\text{msg}}$ then
5:     Return True
6: else
7:     Return False
8: end if
```

Alg. 2 shows the pseudo-code of `checkAuthorship()`, which checks whether or not a message sender possesses the authorship of a claimed EPC. This function is invoked when a message sender (possibly a manufacturer M) claims to be the initial owner of its product. In this case, firstly the message sender's company prefix is retrieved by querying its address in the blockchain. Then, it is compared with the company prefix extracted from the EPC available in the argument. If these prefixes match, it can be confirmed that the message sender is actually the enrolled manufacturer M and the function returns `True`. Otherwise, it simply returns `False`.

2) PRODUCTS MANAGER (`PM`)

In contrast to `MM`, the contract `PM` is created by each manufacturer and consists of four main functions:

1) `enrollProduct()`: Invoked when a manufacturer M first enrolls its own product specified by unique `EPC` and claims its initial ownership;
2) `shipProduct()`: Invoked when a current owner parts with a product and specifies the recipient;
3) `receiveProduct()`: Invoked by the new owner to successfully transfer its ownership when a product is successfully received;
4) `getCurrentOwner()`: Returns the current owner's address.

Algorithm 3 Pseudo-Code of `enrollProduct()` for Enrolling a Product on the Blockchain

```
1: Inputs:
   The PM's address ($A_{\text{msg}}$) and EPC to be enrolled
2: if the message sender has the authorship to claim the initial owner of EPC through checkAuthorship() in PM then
3:     Specify EPC's status as Owned
4:     Specify EPC's owner as $A_{\text{msg}}$
5:     Specify EPC's number of transfer (nTransferred) as 0
6:     Enroll these information on the blockchain
7: else
8:     Do nothing
9: end if
```

Alg. 3 shows the pseudo-code of `enrollProduct()`, which enrolls the manufacturer's information required when its product is stored on the blockchain. Since our POMS restricts that only a single administrator, e.g. `GS1`, can enroll the manufacturer's information, this condition is checked at step 2. If found to be `True`, then the administrator enrolls the manufacturer's information in the blockchain. Note that we assume that an administrator has manually checked that the manufacturer's information is legitimate before sending this transaction. This notion is analogous to the enrollment of a certificate in PKI [30].

Algorithm 4 Pseudo-Code of `shipProduct()` Called When a Product Is Just Left From the Current Owner

```
1: Inputs:
   The message sender's address ($A_{\text{msg}}$), the recipient's address ($A_{\text{rec}}$), and EPC to be transferred
2: if The product with EPC really exists, EPC's status is Owned, and the message sender is the owner of EPC then
3:     Specify EPC's recipient as $A_{\text{rec}}$
4:     Specify EPC's status as Shipped
5:     Enroll these information on the blockchain
6: else
7:     Do nothing
8: end if
```

Alg. 4 shows the pseudo-code of `shipProduct()` which is for the current owner to transfer the product to the next owner. At first, the function checks that the given

EPC exists on the blockchain and the sender of the message actually has the ownership of the EPC. Then, if this is True, shipProduct() specifies the "recipient" as the next recipient's address A_{rec} and EPC's status as Shipped. Note that, at this point in time, the POMS does not change the ownership of the product because of the possibility that this might get lost during the transportation process.

Algorithm 5 Pseudo-Code of receiveProduct() Invoked by a New Owner Who Received a Product

```
1:  Inputs:
    EPC that a new owner received
2:  if the message sender is the recipient of EPC specified by
    the current owner. then
3:      Specify the EPC's owner as $A_{\mathrm{msg}}$
4:      Specify the EPC's status as Owned
5:      Update nTransferred as nTransferred + 1
6:      if nTransferred <= MAXTRANSFER then
7:          Manufacturer $M$ sends incentive as specified by
            transferReward
8:      end if
9:  else
10:     Do nothing.
11: end if
```

Alg. 5 describes the receiveProduct() which is for the receiver to confirm the arrival of the product. The function checks that the claimed EPC is specified by the current owner and that the status of EPC is Shipped. If this is True, the ownership is successfully transferred to the message sender's address. In addition, the manufacturer of the product gives incentive, i.e., some ETH, to the message sender as a reward for obeying the protocol. Since Ethereum requires an execution fee for each transaction, when the current owner sends a product to the recipient, he/she might be reluctant to issue a transaction shipProduct(). To avoid this kind of situation, the following procedure is introduced. If the ownership transfer has been successfully completed, a financial reward transferReward is paid back to the previous owner by the product's manufacturer. The reason why the manufacturer should pay such reward is that in this way counterfeits can be detected and thus identified thanks to their cooperation. It is noted that, in order to avoid the case where two parties repeatedly transfer back and forth to earn rewards, we specify a maximum number of transfers, referred to as MAXTRANSFER. Selecting appropriate values for transferReward and MAXTRANSFER will depend on the actual investment made by the manufacturer for the implementation of POMS. However, such topic is outside the scope of our current research, and thus it will not be considered further.

C. ALGORITHMIC PROCEDURES

Fig. 4 illustrates the detailed system model of the proposed POMS, where two groups of parties are identified. The first

Algorithm 6 Pseudo-Code of getCurrentOwner(), Which Returns the Current Owner's Address of a Product EPC

```
1:  Input: EPC
    Output: the current owner's address of a product EPC
2:  if The product with EPC really exists then
3:      Return the current owner's address of a product EPC
4:  end if
```

one belongs to the supply chain, that is, an administrator, such as GS1, manufacturers, distributors, and retailers. The second group belongs to the post supply chain parties, that is, second hand shops and consumers. Each party possesses Ethereum accounts and manages them with a wallet application operating on personal computers or smartphones/tablets. Next, the procedures on each party in (i) the supply chain and (ii) the post supply chain, will be presented.

1) THE SUPPLY CHAIN

In the supply chain, the following five operational steps are taken.

1) Manufacturer M sends a transaction enrollManufacturer() to a contract MM to enroll its own company prefix specified in EPCs and the company name. Note that this enrollment step requires authentication to avoid counterfeiters from illegally claiming non-authorized company prefix and name.
2) Manufacturer M manufactures N_{products} products and assigns a unique EPC, denoted by EPC_i, for each product i where $1 \leq i \leq N_{\mathrm{products}}$. Simultaneously, to claim the initial ownership of the products, Manufacturer M sends a transaction enrollProduct() to a contract PM to enroll N_{products} EPCs, that is, $(\mathrm{EPC}_1, \mathrm{EPC}_2, \cdots, \mathrm{EPC}_{N_{\mathrm{products}}})$. Since off-the-shelf smartphones and tablets are not equipped with an RFID reader, it is certainly useful and user-friendly that an EPC is also written into a QR code [8] or an NFC (Near Field Communication) tag as proposed in [19].
3) After shipping N_{products} products, manufacturer M issues a transaction shipProduct() to PM for each product to inform distributor D that manufacturer M is now ready to transfer the ownership of the products.
4) When receiving products, distributor D reads the tags' EPCs and checks the genuineness of the EPC. Specifically, for each EPC, distributor D invokes getManufacturerAddress() with EPCs and obtains the manufacturer's address. Then, distributor D invokes getRecipient(), getProductStatus(), and getCurrentOwner() which is shown in Alg. 6. Distributor D verifies the genuineness of the EPC and issues a transaction receiveProduct() to the contract PM with interrogated EPCs, if all of the following conditions are met:
    - The distributor D's address is specified as the recipient;

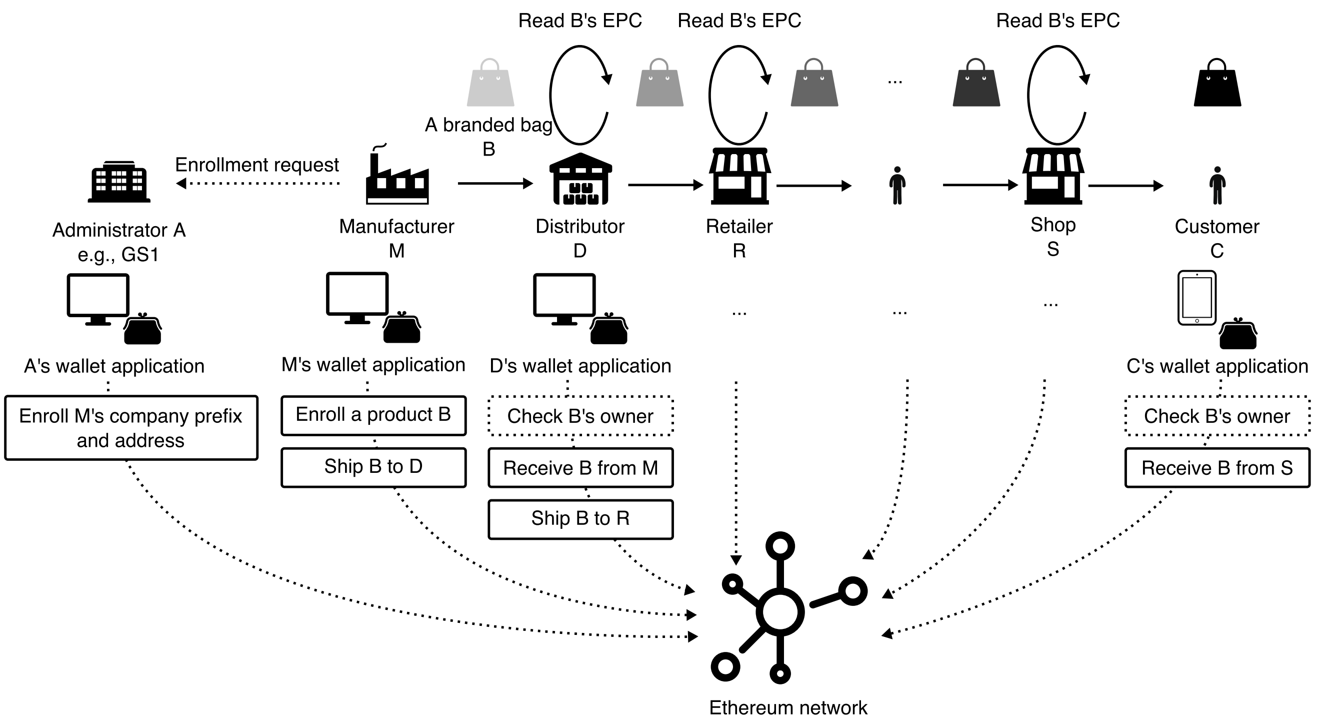


FIGURE 4. Detailed block diagram of the proposed POMS.

- The product status is `Shipped`;
- The address of the current owner is that of a manufacturer M.

The above steps 3 and 4 complete the ownership transfer from manufacturer M to distributor D.

5) If distributor D further transfers products to other parties, i.e., retailers, the same procedure as in the above described step 3 is followed. Similarly, when any party receives products, a recipient follows the same procedure as in step 4.

2) THE POST SUPPLY CHAIN

In the post supply chain, the consumer will decide to buy a product after verifying that the seller actually possesses the ownership of the product. Hence, in contrast to the in the "supply chain situation", an extra step is required before the seller issues a transaction `shipProduct()`. In the following, we assume the situation where consumer C is about to buy a product from its current owner.

1) Consumer C obtains the EPC of the desired product and the current owner's address. Regarding EPC, the buyer can interrogate the EPC of the product via an RFID reader, QR code, or NFC when he/she is physically present in a shop. In contrast, when the buyer cannot physically access the product, e.g. through online shopping, the EPC of the products will be available from a website. As it will be discussed later in Section III-D.1, the current owner must provide a true EPC and his/her address because these information can be verified in the blockchain. Consumer C invokes `getManufacturerAddress()` with the EPC and obtains the manufacturer's address. Then, he/she invokes `getProductStatus()` and `getCurrentOwner()` to manufacturer M's `PM`. If the product status is `Owned`, and the obtained address is the current owner's address, then the buyer will decide to buy the product by paying its selling price.
2) The current owner issues a transaction `shipProduct()` with the EPC of the product to be transferred and the buyer's address.
3) When the product is received, the buyer issues a transaction `receiveProduct()` to the contract `PM` with interrogated EPCs.

D. VALIDATION

In this section, we first explain how counterfeits can be identified and avoided through the operational procedure of the proposed POMS. We then present and discuss various practical issues and cases regarding the actual operation of the POMS.

1) PROTOCOL VERIFICATION

Let us consider all the following possible situations where a party, which can be a new owner of a product, checks the genuineness of the product that he/she will receive from its current owner. Let us assume that this current owner is a counterfeiter and tries to sell/transfer counterfeits to the new owner. For simplicity, in what follows, we denote a current owner and a new owner as a seller and a buyer, respectively.

In this case, three situations are possible:

1) The seller possesses counterfeits with fake EPCs;
2) The seller possesses counterfeits and knows their true EPCs but does not possess their ownership;
3) The seller owns the genuine product and its ownership and possesses a number of its counterfeits too.

For the first situation, the buyer can refuse to buy a counterfeit since he/she can check the product information from its EPC before purchasing it. EPC itself includes product information, e.g. company prefix, item reference, and serial number and thus the party can verify whether or not the EPC is really associated with the product to be purchased.

For the second one, although the buyer is convinced that the EPC is genuinely associated with the desired product, it is doubtful about whether or not the seller possesses the ownership of the product. The buyer first obtains the owner's address of this EPC by querying `getCurrentOwner()`. The seller will then claim that this address is his/her own. Hence, the buyer would like to confirm this claim by making the seller issue a transaction `shipProduct()`, since only the current owner can issue it. However, the seller might refuse to do so before the money is paid. In this case, there is another simple way for the buyer to check on this. After the buyer obtains the current owner's address, he/she generates a challenge message with a pseudo-random number generator and makes the seller sign it. If the signature is verified with the public key that generates the current owner's address, the buyer can be reasonably convinced that the seller is truly the owner of its address. However, in this situation where the counterfeiter is assumed not to possess the ownership of its EPC, the signature verification will surely fail. Hence the buyer can abort this deal.

In the last situation, the buyer is certain that the seller possesses the ownership of the product. In this case, it is possible that a seller ships one of its counterfeits and issues a transaction `shipProduct()` with its EPC. However, we argue that there is no economical merit for the counterfeiter to do so. Since the seller must transfer the ownership of its EPC to the buyer, the original genuine product and any other counterfeits with the same EPCs are no longer sold. Since, in general, counterfeits are much cheaper than the original products, the counterfeiter eventually makes a loss and thus there is no merit for the counterfeiter to do so.

2) MULTIPLE OWNERS CASE

In the previously described protocol validation subsection, it is implicitly assumed that only one party can claim the ownership of one product (EPC). However, it is possible that multiple parties co-possess one product so that the situation gets even more complex than the single owner case. Our POMS can deal with such case by storing multiple owners' addresses for each EPC in `enrollProduct()`. In addition, when the ownership of a product is transferred, a condition must be set beforehand, e.g. with agreements from all owners. This, of course, might be sometimes complex, as for example it is possible that the exact holding ratio is required. In this case, such information must be also stored on the blockchain. Therefore, according to the ownership status, the data structure of owners must be altered.

3) ARBITRATION BETWEEN OWNER AND BUYER

Regarding the transactions, as is often the case with e-commerce, the following unfortunate situation might occur: A buyer pays actual money to the product's owner while he/she does not ship the product and vice versa. In general, this problem is solved by introducing a trusted third party between a buyer and seller, namely *escrow* [31]. However, our POMS does not deal with escrow because it is out of scope in terms of ownership proof. Actually, if a fee is payed by a cryptocurrency, e.g. `ETH`, the escrow is realized without a trusted third party by specifying a clever contract rule so that a cheater loses his/her money [27], [32].

4) OWNER's PRIVACY

One may consider that since any owner address is stored in the `PM`, this could be a privacy issue for customers. However, this issue can be solved by the customers by generating new public/private key pairs for each product. Actually, because this process can be automated by the wallet application of `Ethereum`, the customers do not have to take any precautions.

5) IMPERSONATION AVOIDANCE

One may also consider the situation when an attacker illegally issues transactions by pretending to be a victim, e.g. issuing a transaction under the victim's address. For example, let us consider the following case: Customer C_1 has the ownership of product $\texttt{EPC}_1$ and an attacker wants to steal the ownership of $\texttt{EPC}_1$. In this case an attacker may illegally issue a transaction with `shipProduct()` by specifying its argument as $\texttt{EPC}_1$. However, in `Ethereum`, each transaction must be signed with the sender's private key that also generates his/her address. Therefore, unless a customer leaks his/her private key, nobody else can generate his/her valid signature. Clearly, impersonation can be avoided in this way.

6) CUSTOMER PARTICIPATION

We assume that every party involved with POMS follows the aforementioned protocol. Although it is possible that some parties, like ordinary customers, forget or even might be reluctant to issue `shipProduct()` and `receiveProduct()` transactions, it is fair to assume that all participating parties obey the protocol because of the financial benefit that POMS offers. As previously mentioned and will be emphasized in the next section, the main application of POMS is to be used for dealing with not very cheap products, e.g. branded goods. Hence, the ownership proof of such products should be very much desired by customers.

IV. PERFORMANCE EVALUATION AND DISCUSSION

The proposed POMS has been evaluated in terms of its operational cost. In particular, the total cost has been estimated by measuring the total gas amount for all of the functions involved in the process, that is, (i) enrolling

TABLE 2. Parameters used in the evaluation.

PARAMETER	VALUE
Machine spec	MacBook Pro Early 2015 (OS: macOS 10.11, CPU: 2.7 GHz Intel Core i5, RAM: 16 GB)
`Ethereum` client	`testrpc`
Gas rate [28], [33]	1.443×10^{-6} USD/gas

(`enrollProduct()`), (ii) shipping (`shipProduct()`), and (iii) receiving a product (`receiveProduct()`), and then converting it into a real currency. As the amount of gas is fixed for each operation in `Ethereum`, e.g. an SHA3 calculation costs 20 gas [28], the total gas amount for executing a function is also fixed. Table 2 shows the parameters used in this evaluation. In particular, we have used the `Ethereum`'s test environment tool, `testrpc` [34], to measure the gas amount since this tool has the ability of automatically counting the gas amount. Finally, the total gas amount is converted in USD by referring the conversion chart `CoinGecko` [33]. At the time of evaluation, the rate was 1gas = 0.00001`ETH`= 1.443×10^{-6}USD.

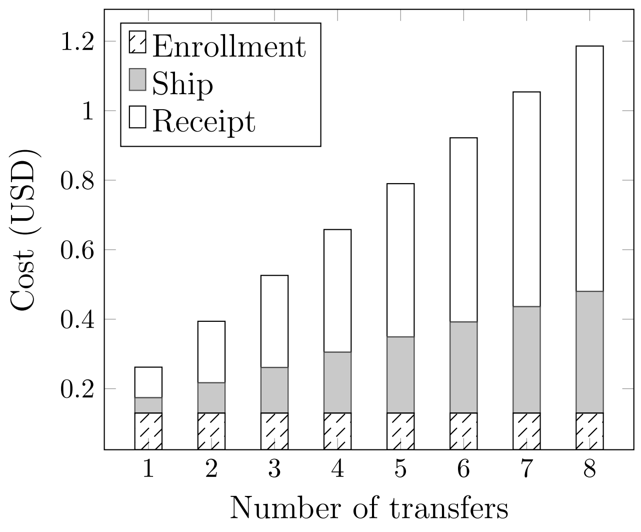


FIGURE 5. Operation cost for an EPC in USD.

Fig. 5 shows the cost, in USD, to manage a product against the number of ownership transfers. Since for each function the gas cost is fixed, the cost increases as the number of ownership transfers increases. However, the obtained result have clearly shown that the total cost remains very low, e.g. for six transfers it is less than US$1. It is noted that this cost is independent of the actual price of the product. Obviously, the proposed POMS is more applicable to relatively expensive products, for example, with selling price more than US$100. Clearly, in practice, relatively inexpensive products are not really worth to be counterfeited.

It is further underlined that the POMS has several advantages as compared to the previous RFID-enabled anti-counterfeits schemes. The first advantage is that even if tags are cloned, any involved party can make decisions as to whether or not products are genuine. Therefore, POMS can detect counterfeits even in the post supply chain scenario. The second advantage is that each party is not required to update the contents of tags for counterfeits detection. This is very important because for other similar RFID-based management systems for tracing products are typically prone to read/write error and they take longer time to write data in the tags [4].

V. OPEN PROBLEMS

There are some open problems which are interesting to be investigated further. The first one relates to how much transfer reward `transferReward` should be paid when the recipient of a product successfully issues the contract `receiveProduct()`. This issue could be addressed by modeling each party's behavior and finding the equilibrium of `transferReward` by game theory. In addition, although we simply introduce a constant `MAXTRANSFER` to avoid the case where any two parties repeatedly transfer ownership back and forth to keep earning rewards, it would be interesting to further investigate other ways to achieve this. For example, (i) gradually decreasing incentive by the number of transfers `nTransferred` and (ii) taking into account the time difference between the last `receiveProduct()` and `shipProduct()` might be possible.

The second topic is related with the issue of security of POMS, especially in connection with the `Ethereum` operation. Since `Ethereum` itself is still under development, its security is not fully verified. In other words, if any critical security issue were found in `Ethereum`, our system would directly be affected. Therefore, it is also an important topic to further verify the security of `Ethereum` together with the proposed POMS.

The third topic relates to the privacy of the manufacturers. That is, since the trace of products is obtained by querying their EPCs to the POMS, sales information of such products could be inferred by the possibly competitors of manufacturers. This means that it is desirable for POMS to possess the following two properties that are difficult to be simultaneously satisfied: (i) *Transparency*, i.e. anyone can check the ownership of products and their traceability, and (ii) *Anonymity*, i.e. the traceability of the products is infeasible while the ownership of products can be proven. In recent years, several cryptocurrencies, which have focused mostly on the anonymity property, have been extensively proposed e.g. [35], [36]. For future works related to POMS, it will be very interesting to investigate how both of these properties can be jointly used for its best compromising operation.

VI. CONCLUSIONS

We have proposed a novel blockchain-based product ownership management system (POMS) for the post supply chain, which makes the efforts of counterfeiters to clone genuine tags redundant since they cannot prove the possession of products on this system. Firstly, the overall practical system requirements have been identified. Then, we have introduced a full-fledged protocol that enables each party, including supply chain partners and customers, to transfer and prove the ownership of RFID tag-attached products. An important advantage of the proposed POMS is that customers can reject the purchase of counterfeits, even with a genuine EPC, under the condition that the seller does not possess their ownership. The protocol validation has shown the validity of our POMS. Based on the proposed protocol, a proof-of-concept experimental system has been implemented on the `Ethereum` platform. Performance evaluation results have shown that the cost for managing products with the proposed POMS is less than US$1 when the number of owner transfers is less than or equal to six.

APPENDIX
KEY CODE SNIPPETS OF OUR POMS

The key code snippets of our implemented contracts, which have been discussed in Section III-B, are listed below. The language used for the code implementation is `Solidity`.

A. MM (ManufacturersManager)

1) `enrollManufacturer()`

The data structure for storing manufacturers' information in `MM`is firstly shown. With such a data structure, the function called `enrollManufacturer()`, which is used for `GS1` to enroll the manufacturer's information in the blockchain, is then described.

```
struct ManufacturerInfo {
  uint40 companyPrefix;
  bytes32 companyName;
  uint expireTime;
}

mapping (address => ManufacturerInfo)
  manufacturers;

mapping (uint40 => address)
  companyPrefixToAddress;
```

```
function enrollManufacturer(address m,
  uint40 companyPrefix, bytes32 companyName,
  uint validDurationInYear) onlyAdmin {
  manufacturers[m].companyPrefix =
    companyPrefix;
  manufacturers[m].companyName =
    companyName;
  manufacturers[m].expireTime = now +
    validDurationInYear;

  companyPrefixToAddress[companyPrefix] =
    manufacturer;
}
```

2) `getManufacturerAddress()`

```
function getManufacturerAddress(uint96 EPC)
  external returns (address) {
  uint40 cp = getCompanyPrefixFrom(EPC);

  return companyPrefixToAddress[cp];
}
```

B. PM (ProductsManager)

1) `enrollProduct()`

`PM` manages the status of each product. Similar to `MM`, a special data structure called `ProductInfo` is firstly shown. Then, the function called `enrollProduct()` is implemented with `ProductInfo` for a manufacturer to claim the first ownership of a product.

```
enum ProductStatus {Shipped, Owned,
  Disposed}

struct ProductInfo {
    address owner;
    address recipient;
    ProductStatus status;
    uint creationTime;
    uint8 nTransferred;
}

mapping (uint96 => ProductInfo) products;
```

```
function enrollProduct(address mmAddr,
  uint96 EPC)
  onlyNotExist(EPC)
  onlyManufacturer {
    ManufacturersManager mm =
      ManufacturersManager(mmAddr);

    if (mm.checkAuthorship(EPC)) {
      products[EPC].owner = manufacturer;
      products[EPC].status =
        ProductStatus.Owned;
      products[EPC].creationTime = now;
      products[EPC].nTransferred = 0;
    }
}
```

2) `shipProduct()` AND `receiveProduct()`

To transfer the ownership, two functions `shipProduct()` and `receiveProduct()` are implemented.

```
function shipProduct(address recipient,
  uint96 EPC) onlyExist(EPC) onlyOwner(EPC)
  onlyStatusIs(EPC, ProductStatus.Owned) {
    if (recipient == products[EPC].owner) {
      throw;
    } else {
      products[EPC].status =
        ProductStatus.Shipped;
      products[EPC].recipient = recipient;
    }
}

function receiveProduct(uint96 EPC)
  onlyExist(EPC)
  onlyRecipient(EPC)
  onlyStatusIs(EPC, ProductStatus.Shipped) {
    products[EPC].owner = msg.sender;
    products[EPC].status =
      ProductStatus.Owned;
    products[EPC].nTransferred =
      products[EPC].nTransferred + 1;
    if (products[EPC].nTransferred
      <= MAXTRANSFER) {
      msg.sender.send(transferReward);
    }
}
```

3) `getCurrentOwner()`

```
function getCurrentOwner(uint96 EPC)
  onlyExist(EPC)
  returns (address) {
  return products[EPC].owner;
}
```

4) `getRecipient()`

```
function getRecipient(uint96 EPC)
  onlyExist(EPC)
  onlyStatusIs(EPC, ProductStatus.Shipped)
  returns (address) {
  return products[EPC].recipient;
}
```

5) `getProductStatus()`

```
function getProductStatus(uint96 EPC)
  onlyExist(EPC) returns (ProductStatus) {
  return products[EPC].status;
}
```

REFERENCES

[1] P. Avery, *The Economic Impact of Counterfeiting and Piracy*. Paris, France: OECD Publishing, 2008. [Online]. Available: http://www.oecd.org/contact/

[2] T. Staake, F. Thiesse, and E. Fleisch, "Extending the EPC network: The potential of RFID in anti-counterfeiting," in *Proc. ACM Symp. Appl. Comput.*, 2005, pp. 1607–1612.

[3] K. Elkhiyaoui, E.-O. Blass, and R. Molva, "CHECKER: On-site checking in RFID-based supply chains," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*, 2012, pp. 173–184.

[4] D. Zanetti, S. Capkun, and A. Juels, "Tailing RFID tags for clone detection," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2013, pp. 15–17.

[5] J. Shi, S. M. Kywe, and Y. Li, "Batch clone detection in RFID-enabled supply chain," in *Proc. IEEE Int. Conf. RFID*, Apr. 2014, pp. 118–125.

[6] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[7] A. Juels, R. Pappu, and B. Parno, "Unidirectional key distribution across time and space with applications to RFID security," in *Proc. USENIX Secur. Symp.*, 2008, pp. 75–90.

[8] *QRcode.Com DENSO Wave*, accessed on Jan. 31, 2017. [Online]. Available: http://www.qrcode.com/en/index.html

[9] (2004). *Combating Counterfeit Drugs, a Report of the Food and Drug Administration*. [Online]. Available: http://www.fda.gov/downloads/Drugs/DrugSafety/UCM169880.pdf

[10] M. Langheinrich and R. Marti, "Practical minimalist cryptography for RFID privacy," *IEEE Syst. J.*, vol. 1, no. 2, pp. 115–128, Dec. 2007.

[11] S. Cai, T. Li, C. Ma, Y. Li, and R. H. Deng, "Enabling secure secret updating for unidirectional key distribution in RFID-enabled supply chains," in *Proc. Inf. Commun. Secur.*, 2009, pp. 150–164.

[12] C. Lv, X. Jia, J. Lin, J. Jing, and L. Tian, "An efficient group-based secret sharing scheme," in *Information Security Practice and Experience*. Berlin, Germany: Springer, 2011, pp. 288–301.

[13] J. G. Alfaro, M. Barbeau, and E. Kranakis, "Proactive threshold cryptosystem for EPC tags," *Ad Hoc Sensor Wireless Netw.*, vol. 12, nos. 3–4, pp. 187–208, 2011.

[14] K. Toyoda and I. Sasase, "Secret sharing based unidirectional key distribution with dummy tags in Gen2v2 RFID-enabled supply chains," in *Proc. IEEE Int. Conf. RFID*, Apr. 2015, pp. 84–90.

[15] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[16] D. Zanetti, L. Fellmann, and S. Capkun, "Privacy-preserving clone detection for RFID-enabled supply chains," in *Proc. IEEE Int. Conf. RFID*, Sep. 2010, pp. 37–44.

[17] E.-O. Blass, K. Elkhiyaoui, and R. Molva, "Tracker: Security and privacy for RFID-based supply chains," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2011, pp. 1–20.

[18] J. Huang, X. Li, C. Xing, W. Wang, K. Hua, and S. Guo, "DTD: A novel double-track approach to clone detection for RFID-enabled supply chains," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 134–140, Jan. 2015.

[19] M. Q. Saeed, Z. Bilal, and C. D. Walter, "An NFC based consumer-level counterfeit detection framework," in *Proc. IEEE Int. Conf. Privacy, Secur. Trust (PST)*, Sep. 2013, pp. 135–142.

[20] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, Apr. 2004.

[21] Everledger. *Everledger | A Digital Global Ledger*. Accessed on Aug. 1, 2017. [Online]. Available: https://www.everledger.io/

[22] Blockverify. *Block Verify*. Accessed on Aug. 1, 2017. [Online]. Available: http://blockverify.io/

[23] F. Charlon. (Dec. 2013). *Openassets/Open-Assets-Protocol: Technical Specification for the Open Assets Protocol, a Bitcoin Based Colored Coins Implementation*. [Online]. Available: https://github.com/OpenAssets/open-assets-protocol

[24] M. Rosenfeld, "Overview of colored coins," White Paper, 2012. [Online]. Available: https://bitcoil.co.il/BitcoinX.pdf

[25] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7467408

[26] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014.

[27] Ethereum. (2016). *Solidity Documentation*. [Online]. Available: https://ethereum.github.io/solidity/docs/home/

[28] *Gas Fees for Ethereum Operations*, accessed on Dec. 10, 2016. [Online]. Available: http://ether.fund/tool/gas-fees

[29] A. Loibl and J. Naab, "Namecoin," in in *Proc. Seminars Future Internet (FI) Innov. Internet Technol. Mobile Commun. (IITM)*, Munich, Germany, Sep. 2014, pp. 107–113. [Online]. Available: https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2014-08-1.pdf

[30] P. R. S. Boeyen and T. Howes, *Internet X.509 Public Key Infrastructure Operational Protocols—LDAPv2*, document RFC 2559, Apr. 1999. [Online]. Available: https://tools.ietf.org/html/rfc2559

[31] *What is Escrow? how Does Escrow Work?—Escrow.com*, accessed on Jan. 31, 2017. [Online]. Available: https://www.escrow.com/what-is-escrow

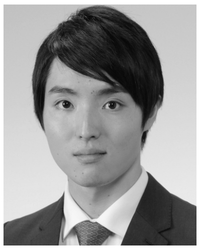
[32] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, "Step by step towards creating a safe smart contract: Lessons and insights from a Cryptocurrency Lab," in *Financial Cryptography and Data Security*, J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds. Berlin, Germany: Springer, 2016, pp. 79–94, doi: 10.1007/978-3-662-53357-4_6.

[33] *Ethereum Price Chart (ETH/USD)|Coingecko*, accessed on Oct. 21, 2016. [Online]. Available: https://www.coingecko.com/en/price_charts/ethereum/usd

[34] EthereumJS. *GitHub—Ethereumjs/testrpc: Fast Ethereum RPC client for testing and development*. Accessed on Aug. 1, 2017. [Online]. Available: https://github.com/ethereumjs/testrpc

[35] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed E-Cash from Bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, Oct. 2013, pp. 397–411.

[36] E. Ben Sasson *et al.*, "Zerocash: Decentralized anonymous payments from Bitcoin," in *Proc. IEEE Symp. Secur. Privacy (SP)*, Apr. 2014, pp. 459–474.



KENTAROH TOYODA (M'16) was born in Tokyo, Japan, in 1988. He received the B.E., M.E., and Ph.D. degrees in engineering from Keio University in 2011, 2013, and 2016, respectively. He is an Assistant Professor with Keio University. His research interests include security and privacy for systems and services with Internet of Things devices and financial technology. He is a member of IEICE and IPSJ. He was a recipient of the Fujiwara Foundation Award in 2016, the Telecom System Technology Encouragement Award in 2015, and the IEICE Communication Society Encouragement Awards in 2012 and 2015.

P. TAKIS MATHIOPOULOS (SM'94) received the Ph.D. degree in digital communications from the University of Ottawa, Ottawa, ON, Canada, in 1989. From 1982 to 1986, he was with Raytheon Canada Ltd., where he was involved in air navigational and satellite communications. In 1989, he joined the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada, where he was a Faculty Member until 2003, holding the rank of a Professor from 2000 to 2003. From 2000 to 2014, he was with the Institute for Space Applications and Remote Sensing (ISARS), National Observatory of Athens, where he established the Wireless Communications Research Group. As ISARS Director (2000–2004), he led the Institute to a Significant Expansion Research and Development Growth and International Scientific Recognition. For these achievements, ISARS has been selected as a National Centre of Excellence for the years 2005–2008. From 2008 to 2013, he was a Guest Professor with Southwest Jiaotong University, Chengdu, China. He has been also appointed by the Government of China as a Senior Foreign Expert with the School of Information Engineering, Yangzhou University, Yangzhou, China, from 2014 to 2017, and by Keio University as a Guest Professor (Global) with the Graduate School of Science and Technology, from 2015 to 2017, under the Top Global University Project of the Ministry of Education, Culture, Sports, Science and Technology of the Government of Japan. Since 2014, he has been a Professor of Telecommunications with the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece. In his research areas, he has co-authored over 120 journal papers mainly published in various IEEE and IET journals, one book, four book chapters, and over 140 conference papers. For the last 25 years, he has been conducting research mainly on the physical layer of digital communication systems for terrestrial and satellite applications, as well as in the fields of remote sensing, LIDAR systems, and the Internet of Things.

He has been a member of the Technical Program Committees (TPC) of over 70 international IEEE conferences, and TPC Vice Chair and Co-Chair of several IEEE and other influential conferences. He was a recipient of the Advanced Systems Institute Fellowship and the Killam Research Fellowship, and a co-recipient of two best paper awards. He has been or currently serves on the Editorial Board of several archival journals, including the IET Communications and the IEEE TRANSACTIONS ON COMMUNICATIONS (1993–2005). He has regularly served as a Consultant for various governmental and private organizations. Since 1993, he has been serving on a regular basis as a Scientific Advisor and a Technical Expert for the European Commission (EC). Furthermore, from 2001 to 2014, he has served as a Greek Representative to high-level committees in the EC and the European Space Agency. He has delivered numerous invited presentations, including plenary and keynote lectures, and has taught many short courses all over the world.

IWAO SASASE (SM'03) was born in Osaka, Japan, in 1956. He received the B.E., M.E., and D.Eng. degrees in electrical engineering from Keio University, Yokohama, Japan, in 1979, 1981, and 1984, respectively. From 1984 to 1986, he was a Post-Doctoral Fellow and a Lecturer of Electrical Engineering with the University of Ottawa, ON, Canada. He is currently a Professor of Information and Computer Science with Keio University. He has authored over 280 journal papers and 420 international conference papers. He granted 43 Ph.D. degrees to his students in his research fields. His current research interests include modulation and coding, broadband mobile and wireless communications, optical communications, communication networks, and information theory.

He is a fellow of IEICE and a member of the Information Processing Society of Japan. He received the 1984 IEEE Communications Society (ComSoc) Student Paper Award (Region 10), the 1986 Inoue Memorial Young Engineer Award, the 1988 Hiroshi Ando Memorial Young Engineer Award, the 1988 Shinohara Memorial Young Engineer Award, the 1996 Institute of Electronics, Information, and Communication Engineers (IEICE) of Japan Switching System Technical Group Best Paper Award, and the WPMC 2008 Best Paper Award. He served as the President-in-Elect of the IEICE ComSoc (2012–2013). He was the Chair of the Satellite and Space Communications Technical Committee (2000–2002) of IEEE ComSoc, the Director of the Society of Information Theory and its Applications in Japan (2001–2002), the Chair of the Communication System Technical Committee (2002–2004) of the IEICE ComSoc, the Director of the Asia Pacific Region (2004–2005), the Vice President of the Communications Society (2004–2006), the Chair of the Network System Technical Committee (2004–2006), the Board of Governors Member-at-Large (2010–2012), and the Japan Chapter Chair (2011–2012).

TOMOAKI OHTSUKI (SM'01) received the B.E., M.E., and Ph.D. degrees in electrical engineering from Keio University, Yokohama, Japan, in 1990, 1992, and 1994, respectively. From 1994 to 1995, he was a Post-Doctoral Fellow and a Visiting Researcher of Electrical Engineering with Keio University. From 1993 to 1995, he was a Special Researcher of Fellowships of the Japan Society for the Promotion of Science for Japanese Junior Scientists. From 1998 to 1999, he was with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, USA. From 1995 to 2005, he was with the Tokyo University of Science. In 2005, he joined Keio University. He is currently a Professor with Keio University. He has authored or co-authored over 140 journal papers and 340 international conference papers. He is involved in research on wireless communications, optical communications, signal processing, and information theory.

He is a fellow of the IEICE. He was a recipient of the 1997 Inoue Research Award for Young Scientist, the 1997 Hiroshi Ando Memorial Young Engineering Award, the Ericsson Young Scientist Award 2000, the 2002 Funai Information and Science Award for Young Scientist, the IEEE the 1st Asia-Pacific Young Researcher Award 2001, the 5th International Communication Foundation Research Award, the 2011 IEEE SPCE Outstanding Service Award, the 28th TELECOM System Technology Award, the ETRI Journals 2012 Best Reviewer Award, and the 9th International Conference on Communications and Networking in China 2014 (CHINACOM '14) Best Paper Award. He gave tutorials and keynote speech at many international conferences, including the IEEE VTC, the IEEE PIMRC, and so on. He is currently serving as a Vice President of the Communications Society of the IEICE. He served as the Chair of the IEEE Communications Society and the Signal Processing for Communications and Electronics Technical Committee. He served as a Technical Editor of the *IEEE Wireless Communications Magazine*. He is currently serving an Editor of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS and *Elsevier Physical Communications*. He has served General Co-Chair and Symposium Co-Chair of many conferences, including the IEEE GLOBECOM 2008, SPC, the IEEE ICC 2011, CTS, and the IEEE GLOBECOM 2012, SPC.