IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 November 2022, accepted 17 November 2022, date of publication 23 November 2022,
date of current version 30 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3224230

RESEARCH ARTICLE

Dispute Resistance Multilayered RFID Partial Ownership Transfer With Blockchain

MING-HOUR YANG[1], (Member, IEEE), YU-SHAN HSU[2], AND HUNG-YU KO[1]
[1]Department of Information and Computer Engineering, Chung Yuan Christian University, Taoyuan City 320314, Taiwan
[2]Department of Electronic Engineering, Chung Yuan Christian University, Taoyuan City 320314, Taiwan

Corresponding author: Ming-Hour Yang (mhyang@cycu.edu.tw)

This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST 111-2221-E-033-047, Grant MOST 111-2218-E-A49-013-MBK, and Grant MOST 111-2218-E-002-038; and in part by the National Defense Science and Technology Academic Collaborative Research Project, in 2022.

ABSTRACT Frontend supply chains involve the transfer of large quantities of products. Therefore, incomplete transfer and product loss are inevitable. This paper presents a high-efficiency group ownership transfer protocol that incorporates blockchains to enable the efficient transfer of large quantities of products and prevent disputes regarding transfer completion. This protocol collocates authenticated information of a manufacturer with its exclusive blockchain address to generate an initial transaction block. The blockchain is then used to trace the original product manufacturers to prevent counterfeiting, and a grouping proof is employed to prevent disputes regarding transfer comprehensiveness. The proposed protocol was capable of resisting common off-chain radio frequency identification ownership transfer attacks, indicating its security. A security testing tool was employed to prove whether the on-chain smart contracts could also resist the attacks on blockchain. The analysis and experiment results revealed that the proposed protocol required fewer messages for ownership transfer and half the calculation time compared with the existing protocols. Thus, for the simultaneous transfer of ownership of a massive number of tags, the proposed protocol is the most efficient, with a reduced calculation time, thus making it most suitable for bulk cargo ownership transfer in the frontend supply chain environment.

INDEX TERMS Blockchain, multilayered ownership transfer, supply chain management, counterfeit RFID tag attack.

I. INTRODUCTION

According to the prediction by Statista, e-commerce is flourishing [1]. In 2020, the global retail e-commerce sales amounted to US$4.28 trillion in 2020 and are predicted to grow to US$5.42 trillion in 2022 at a growth rate of 26%. To improve cargo management efficiency, product supply chains began to apply radio frequency identification (RFID) tags on products for automatic inventory and ownership management.

Ownership transfer in a product supply chain is carried out at the backend as well as the frontend [2], [3]. Retailers and consumers in the backend supply chain transfer the ownership of only a few products each time. Because of privacy concerns related to individuals' consumption habits and purchase of sensitive products, studies on single-tag ownership transfer [4], [5], [6], [7], [8], [9], [10] have focused on security and privacy issues but have overlooked problems associated with transfer efficiency [3]. By contrast, manufacturers and wholesalers in the frontend supply chain may transfer the ownership of a large quantity of products each time. Therefore, group tag transfer protocols have been proposed to enhance the efficiency of bulk cargo ownership transfer [11], [12], [13], [14], [15], [16], [17], [18], [19]. However, because of the sheer quantity of products, preventing incomprehensive transfer, accidents, or employee theft during transportation is difficult, which may hinder the successful delivery of the products to their destinations, incurring disputes. Theft that occurs in

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

the supply chain, with up to US$30 billion being lost each year [20]. On average, participating retailers attributed the greatest portion of losses (33.2%) to external theft, followed by internal employees [21].

Consumers and government worldwide have begun to emphasize traceability [22] to ensure that their products come from desired manufacturers. Therefore, product supply chains have employed RFID tags on products to record production, storage, and sales information in all stages in a supply chain. Consumers read these tags and acquire product information through an object name server, which helps them identify counterfeit or pirated products [23]. However, when these tags are displayed in public, they are vulnerable to attackers' duplication. Moreover, conventional business models rely on centralized trusted third-party servers. If these servers do not have comprehensive security mechanisms such as communication standards, attackers can still pose as authorized agencies or incur man-in-the-middle attacks on servers such as ONS [24]. Consequently, consumers purchase counterfeit or pirated products because of false information. To satisfy both consumers' needs for product traceability and information security, the production and sales history of products is recorded in distributed ledger technology to prevent consumers from acquiring incorrect transaction information [25].

To secure information privacy and traceability of a product within its life cycle and make it suitable for bulk cargo transfers in a frontend supply chain, a hierarchical mobile RFID structure was constructed in this study by applying multiple mobile readers. A high-efficiency group ownership transfer method integrated with blockchains was introduced. This method enables the off-chain ownership transfer of products with RFID tags and the on-chain tracing of the ownership transfer information. The information was consistent between off-chain and on-chain ownership. In addition, the proposed method enables the management of ownership and product manufacturing information through smart contracts. Product ownership transfer history is recorded in blockchains, which are unchangeable. Thus, blockchains can be used to trace the original manufacturers and prevent product counterfeiting. The ownership transfer protocol incorporates grouping proofs to prevent disputes on product transfer comprehensiveness. The major contributions are as follows: (1) the protocol enables the transfer of a large of tags at one time through a multilayered reader securely; (2) the protocol is capable of cross-authority ownership transfer; (3) the protocol enables partial ownership transfer, wherein the ownership of one or more RFID tags is transferred at one time; (4) the protocol verifies the comprehensiveness of product transactions through grouping proofs and enables the clarification of the attribution of responsibility when disputes occur; (5) the protocol prevents known RFID tagged ownership transfer attacks such as replays, eavesdropping, and tag counterfeiting; (6) the protocol enables the management of multiple manufacturers through a single contract to reduce blockchain transaction costs. The results of an analysis revealed that the proposed protocol requires lower message and computational load than the existing group ownership transfer protocols with grouping proofs.

The remainder of the paper is organized as follows. Section 2 reviews the previously literature. Section 3 presents the environmental assumptions considered in this study and the structure of the multilayered reader incorporating a blockchain. Section 4 describes the protocol and smart contract proposed in this study, which are used to initialize and transfer ownership as well as update keys. Section 5 describes the security analysis of common RFID attacks and the comparison of security between the proposed protocol and the existing group ownership transfer protocols. Section 6 presents a comparison of the efficacy of the proposed protocol with that of the existing group ownership transfer protocols. Section 7 concludes this study and proposes future directions.

II. RELATED WORK

To clearly attribute responsibilities for lost products, Juels et al. [26] proposed the Yoking-proof protocol, in which a grouping proof is generated during the transfer of products between the original and new owners to ensure that the products are successfully and completely delivered; this prevents both parties from repudiating the completed transactions and product delivery. Nevertheless, attackers can initiate replay attacks by replaying some of the grouping proofs to generate comprehensive, legal grouping proof without the actual products in the transaction. Saito and Sakurai [27] proposed grouping proofs based on timestamps to prevent replay attacks; however, attackers can also control timestamps to initiate replay attacks. Piramuthu [28] employed random numbers to develop a protocol that ensures information freshness of grouping proof. Nevertheless, Lopez et al. [29] point out that attackers can eavesdrop the original grouping proof information and replace partial sessions of the grouping proofs to generate legal grouping proofs, which may create an inconsistency between the transferred products and the original products. Lopez et al. [29] proposed a grouping proof protocol that considers such multiple proof attacks.

However, the efficacy of this protocol is limited because conventional grouping proofs are generated according to a specific order [26], [27], [28], [29], [30], [31], [32], [33]. Because generating and recombining grouping proofs one by one is time-consuming, computing methods that do not require waiting sequences, such as broadcast messages and exclusive OR (XOR), can be employed to shorten the time required to generate a grouping proof [34], [35], [36], [37], [38], [39], [40], [41]. However, when tags return messages simultaneously and anticollision algorithms such as tree-walker and Aloha are used to stagger message response time, attackers can utilize the time difference and generate grouping proofs from tags that are not generated at the same time points [42]. Furthermore, when the number of tags exceeds the maximal number a reader can read and must be read in batches, the overall reading time exceeds the threshold value,

preventing the generation of legal grouping proofs [43]. Yang et al. [15] proposed a hierarchical group tag transfer mechanism in which multiple readers read tags simultaneously, thereby overcoming the reading limitations and effectively preventing known attacks against RFID. However, when the grouping proof protocol, which safeguards comprehensive product delivery, and the tag group transfer protocol, which performs ownership transfer, are executed separately, transfer efficacy is impeded [44], [45]. Tsai et al. [44] combined these two protocols for identity authentication and random number generation and proposed an ownership transfer protocol that incorporates grouping proofs, which safeguarded both the efficacy and security of tag group ownership transfer.

In order to guaranteed the genuineness of RFID tags in the post supply chain, Toyoda et al. [2] proposed a blockchain-based ownership management system in which product and owner information is signed and recorded in a blockchain. Consumers can acquire the proof of possession of products through the blockchain. Thus, attackers who counterfeit product tags cannot modify the transaction history of the products in the blockchain and cause consumers to acquire incorrect products. Nevertheless, attackers may still acquire an owner's true identity through the blockchain address of a transaction, leading to privacy risks [46]. Kosba et al. [47] proposed Hawk, a smart contract that applies zero-knowledge proofs to protect blockchain data privacy. However, attackers can learn about the direction of commerce for a product by tracking its electronic product code (EPC). The EPCglobal standard [48] has been implemented to protect consumers' privacy by hiding tag codes, but attackers can still acquire these codes by using noncompliant readers [49].

III. MULTILAYERED OWNERSHIP TRANSFER METHOD RESISTANT TO OWNERSHIP TRANSFER DISPUTES

The proposed ownership transfer method incorporates off-chain ownership transfer of products with RFID tags and on-chain ownership transfer history and is suitable for bulk cargo transfer in a frontend supply chain. It employs a multilayered mobile RFID reader to read a large number of tags simultaneously, thereby improving transfer efficacy. To implement our scheme, we leverage Ethereum, a blockchain-based consensus platform that enables the integration of product ownership into tags and blockchains.

A. PRELIMINARIES

As depicted in Fig. 1, the supply chain composed of five actors, i.e. manufacturers, warehouses, wholesalers, retailers and customers. The proposed logistic ownership transfer system consists of off-chain and on-chain blocks. In the upper area encircled with bold lines in Fig. 1 is off-chain area, Under the assumption that each supply chain actor has possess a backend server and multiple mobile RFID readers. The supply chain actor is able to reads multiple RFID tags simultaneously by using multiple readers which authorized by their own backend server for product ownership transfer.

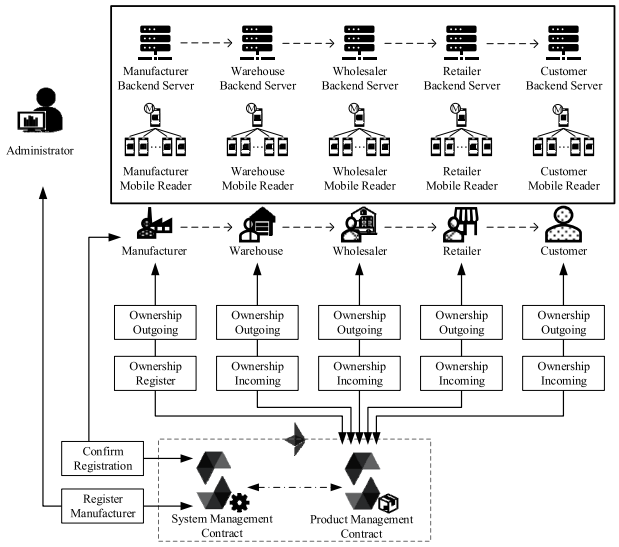


FIGURE 1. On-chain and off-chain logistics ownership transfer system frameworks.

The backend server is used to manage the actor's own tag key and serves as a node in the blockchain for running smart contracts. The main reader is marked with M, and those that are unmarked are auxiliary readers. The auxiliary readers receive the property ownership transfer messages transmitted by the main readers and further transmit them to the RFID tags of the products allocated to the auxiliary readers, thereby assisting in ownership transfer and the generation of grouping proofs.

The lower area, which is on chain area, encircled with dotted lines features two on-chain smart contracts, one namely the system management contract, which is used to authenticate manufacturers, and the other namely product management contract, which records the ownership transfer history of each product.

The manufacturers need to register their information through the system management contract before selling their products. In order to avoid any non-authorized actors from illegally issuing the ownership of products, the administrator verifies the correctness of the on-chain manufacturer data and authorizes the manufacturers to access the product management contract. (GS1 [50] is a suitable administrator. Currently, manufacturers applying RFID for logistics management have registered their data on GS1).

Next, the supply chain actors can use the product management contract to transfer products and receive products

The remainder of this section describes the on-chain manufacturer and product registration procedure and the off-chain environmental assumptions.

B. ON-CHAIN ENVIRONMENTAL ASSUMPTIONS AND REGISTRATION

In the proposed ownership transfer system, in order to sell their products, manufacturers are required to register their information in the blockchain. Assume Manufacturer A uses

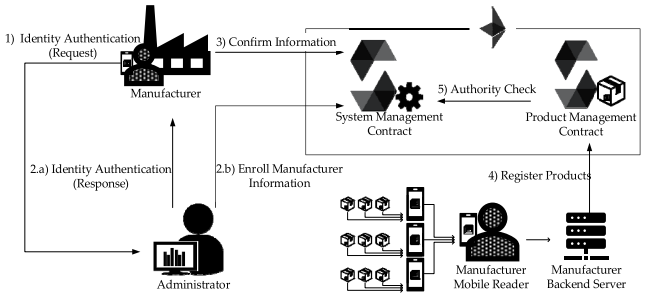


FIGURE 2. Manufacturer registering its products.

a secure channel to submit its blockchain address $Addr_A$ and manufacturer data $Data_A$ (e.g., name, factory addresses, and contacts) to the administrator for identity authentication (Step 1, Fig. 2). After the identity and data are verified to be correct, the administrator issues an EPC prefix PRC_A to manufacturer A and submits $Addr_A$, $Data_A$, and PRC_A to the system management contract (Step 2, Fig. 2). After receiving PRC_A, Manufacturer A also submits $Addr_A$, $Data_A$, and PRC_A to the system management contract (Step 3, Fig. 2). After the system management contract confirms that the data submitted by Manufacturer A and the administrator are consistent, the contract records $Addr_A$, $Data_A$, and PRC_A and authorizes Manufacturer A to register their products through the product management contract.

After Manufacturer A receives the authorization to register a product, he/she can use $Addr_A$ to generate a transaction message with a product tag identification (ID) code through the backend server and submit it to the product management contract for registration. After the contract receives the message and confirms with the system management contract that the code can be registered in $Addr_A$, it assigns $Addr_A$ as the original owner of the product. The symbols used in this study and their definitions are presented in Table 1.

C. OFF-CHAIN ENVIRONMENTAL ASSUMPTIONS AND METHODS

Product tag ID are recorded in the blockchain to integrate the product ownership in the tag and blockchain. During ownership transfer, to ensure the consistency between the on-chain and off-chain transaction products, the ownership of the blockchain and RFID tag of a product must be transferred simultaneously. Because the volumes of logistics frontend products are large, a group key is established on the backend server. A single multicast message is transmitted to all the tags belonging to the same group to reduce the number of times the message is transferred. However, when the number of tags exceeds the upper limit of the messages a reader can read [43], the tags must be read using more than one message. Therefore, a multilayered reader is employed in this study to enable the main readers to distribute messages to their own auxiliary readers, thereby controlling the number of tags each reader must process below the upper limit. Thus, tags can be read simultaneously using all readers.

TABLE 1. Symbol definitions.

B	Blockchain
ori	Original owner
new	New owner
$Data_e$	Manufacturer data of owner, e
PRC_e	EPC prefix of the RFID tag of owner, e
$Addr_e$	Blockchain address owned by owner, e
R_e^x	The x-th reader owned by owner, e
R_e	Set of the m readers owned by owner, e; $R_e = \{R_e^1, R_e^2, \ldots, R_e^m\}$
RID_e^x	ID of the x-th reader (R_e^x) owned by owner, e
RID_e	ID set of the m readers owned by owner, e; $RID_e = \{RID_e^1, RID_e^2, \ldots, RID_e^m\}$
T_e^i	Tag of the i-th product owned by owner, e
T_e	Tag set of the n products owned by owner, e; $T_e = \{T_e^1, T_e^2, \ldots, T_e^n\}$
TID_e^i	ID of the i-th product tag (T_e^i) owned by owner, e
TID_e	ID set of the n product tags owned by owner, e; $TID_e = \{TID_e^1, TID_e^2, \ldots, TID_e^n\}$
D_e	Backend server owned by owner, e
DID_e	ID of the backend server owned by owner, e
G_e^p	Node of the p-th tag group governed by owner, e
GK_e^p	Group key of the node of the p-th tag group (G_e^p) governed by owner, e
r_0	Random number generated by the backend server of the original owner
r_1, r_4	Random number generated by the backend server of the new owner
r_2	Random number generated by the product management contract
r_3^i	Random number generated by product tag, T_e^i
r_3	Set of random numbers generated by product tag, r_3^i, generated by product tag T_e^i; $r_3 = \{r_3^1, r_3^2, \ldots, r_3^n\}$
DK	Key shared by the backend servers of the original and new owners
PB_e	Public key owned by the server of owner, e
PV_e	Private key owned by the server of owner, e
RK_e	Key shared between the main mobile reader and backend server of owner, e
MK_e^m	Key shared between the main mobile reader and the m-th auxiliary reader of owner, e
MK_e	Set of keys shared between the main mobile reader and the m auxiliary readers of owner, e; $MK_e = \{MK_e^1, MK_e^2, \ldots, MK_e^m\}$
TK_e^i	Key shared by the product tag T_i and backend server owned by owner, e
TK_e^i	Key shared by the product tag T_i and backend server owned by owner, e
TK_e	Set of keys shared by the n product tags (T_i) and backend server owned by owner, e; $TK_e = \{TK_e^1, TK_e^2, \ldots, TK_e^n\}$
$E(key, msg)$	Symmetric encryption and decryption algorithm; a key (key) is used to encrypt a message (msg)
$Sign(key, msg)$	Signature for using key to generate msg
$H(msg)$	Hash value of msg
$MAC(key, msg)$	Message authentication code (MAC) for the msg generated using key

To enable backend servers to generate multicast messages to target tags and reduce the number of tags each reader is

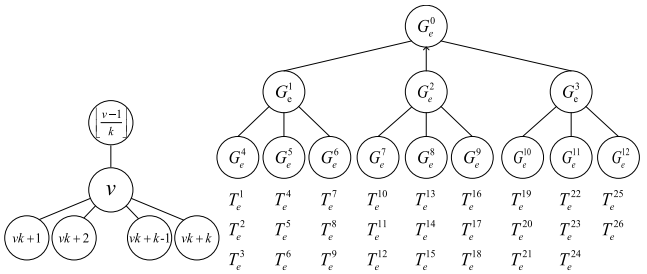


FIGURE 3. Tag group tree: (a) group node coding rules; (b) example of the coding of tag group nodes and their tags.

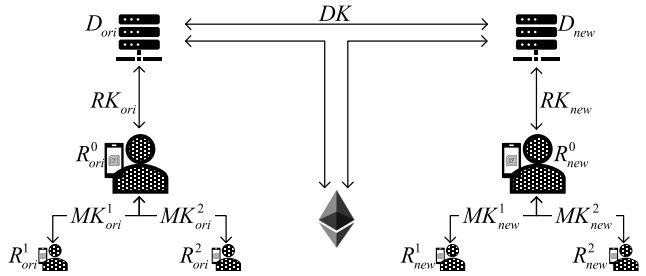


FIGURE 4. Keys shared by the owners' servers and readers.

required to read, the tags must be grouped, and appropriate group keys must be generated accordingly. In this study, n tags are grouped to generate a k-ary group tag tree with the height of $\lceil \log_k \frac{n}{k} \rceil + 1$. Each subtree differs from the tree in heights no greater than 1. See Fig. 3(a) for the rules of coding each group node. The nodes are coded in order from top to bottom and from left to right. When the code of a node is v, its parent and child nodes are coded $\left\lfloor \frac{v-1}{k} \right\rfloor$ and $v * k + 1, v * k + 2, \ldots, v * k + k$, respectively. In (1), the $\lceil n/k \rceil$ group nodes coded from $\left\lceil \frac{(n/k)-1}{k-1} \right\rceil$ to $\left\lceil \frac{(n/k)-1}{k-1} \right\rceil + \lceil n/k \rceil - 1$ and directly connected to a tag are defined as leaf group nodes ($G_e^{leaf,q}$). Fig. 3(b) presents a 3-ary ($k = 3$) group tag tree consisting of 26 tags ($n = 26$). According to (1), the tags T_e^1, T_e^2, and T_e^3 are connected to the leaf group node $G_e^{leaf,1}$ ($q = 1$). In other words, the first group node is G_e^4. Furthermore, the nodes from T_e^1 to T_e^9 are grouped into the nodes G_e^4, G_e^5, and G_e^6 and belong to the parent group node G_e^1.

$$G_e^{leaf,q} = \left\{ T_e^i | \forall i T_e^i \in G_e^{leaf,q}, (q-1)\,k + 1 \le i \le qk, \; 1 \le q \le \lceil n/k \rceil \right\} \tag{1}$$

Assume each tag T_e^i has a unique tag ID (TID_e^i) and shares with its owner's backend server the tag key TK_e^i [51], and the group key GK_e^q is calculated using the keys shared by all tags under the group node G_e^p. If the tag T_e^i belongs to the group node G_e^p, then T_e^i can decrypt the messages encrypted by the group key GK_e^q. As shown in Fig. 3(b), the tags T_e^1, T_e^2, and T_e^3 belong to G_e^4, and tag keys TK_1, TK_2, and TK_3 can be used to generate the group key GK_e^4.

To ensure security of message transmission, the backend servers and readers are hypothesized to share keys (Fig. 4). The backend servers of the original (*ori*) and new (*new*) owners share the key DK. The backend server of the original owner (D_{ori}) shares with its main reader (R_{ori}^0) the key RK_{ori}. The m-th auxiliary reader (R_{ori}^m) shares with the main reader (R_{ori}^0) the key MK_{ori}^m. Similarly, the backend server of the new owner (D_{new}) shares with its main reader (R_{new}^0) the key RK_{new}. The s-th auxiliary reader (R_{new}^s) shares with the main reader (R_{new}^0) the key MK_{new}^s.

IV. RFID TAG OWNERSHIP TRANSFER PROTOCOL

For the traceability of ownership transfer, smart contracts are applied to record ownership outgoing and incoming. The records are integrated with the off-chain RFID tags of actual products. A multilayered mobile RFID reader is implemented to simultaneously read a large number of tags to enhance transfer efficiency. The process of ownership transfer is divided into ownership outgoing and incoming.

A. OWNERSHIP OUTGOING

When *ori* transfers its tag set T_{ori} to *new* (Fig. 5), the main reader of *ori* (R_{ori}^0) binds the transfer message (OT), random number (r_0), the ID of the main reader of *new* (RID_{new}^0), and the ID set of the transferred tags (TID_{ori}) with the outgoing tag for *new*. The key RK_{ori} shared between R_{ori}^0 and the server D_{ori} is then used to encrypt the message M_1 to D_{ori}.

After D_{ori} receives the message M_1, it generates the random number r_0 and integrates it into the decrypted M_1. The message is then encrypted with DK, the key shared by D_{ori} and D_{new}. The message M_2 is then transmitted from D_{ori} to D_{new}.

After D_{new} receives and decrypts M_2 and authenticates its source, it generates the random number r_1 and creates new key $TK_{new}^{i\,\prime}$ for each tag TK_{ori}^i in TID_{ori}. The key is used to encrypt the ID of each incoming tag TID_{ori}^i. Subsequently, D_{new} provides the hash values used by the blockchain to verify all transferred tags ($\mathrm{H}\left(TK_{new}^{i\,\prime} \oplus r_1\right)$), the blockchain address used for receiving the products ($Addr_{new}$), and random number r_0 to D_{ori}.

After D_{ori} decrypts the message M_3 from D_{new}, confirms r_0 as the random value for the transaction, and authenticates the source of the message, it employs the outgoing function SingOut()in the product management contract and its blockchain address $Addr_{ori}$ to set up all the TID_{ori}^i of the tag TID_{ori} to the outgoing status. When the product management contract confirms that the owner possesses the ownership of TID_{ori}, it generates the random number r_2 for completion of status setting.

After D_{ori} receives r_2 from the product management contract (Fig. 6), it combines M_3 from D_{new} with r_2 and generates an outgoing message $M_{update}^i = \mathrm{E}(TK_{ori}^i, TID_{ori}^i \parallel \mathrm{H}\left(TK_{new}^{i\,\prime} \oplus r_1\right) \parallel r_2) \parallel TID_{ori}^i$ for each product tag T_{ori}^i.

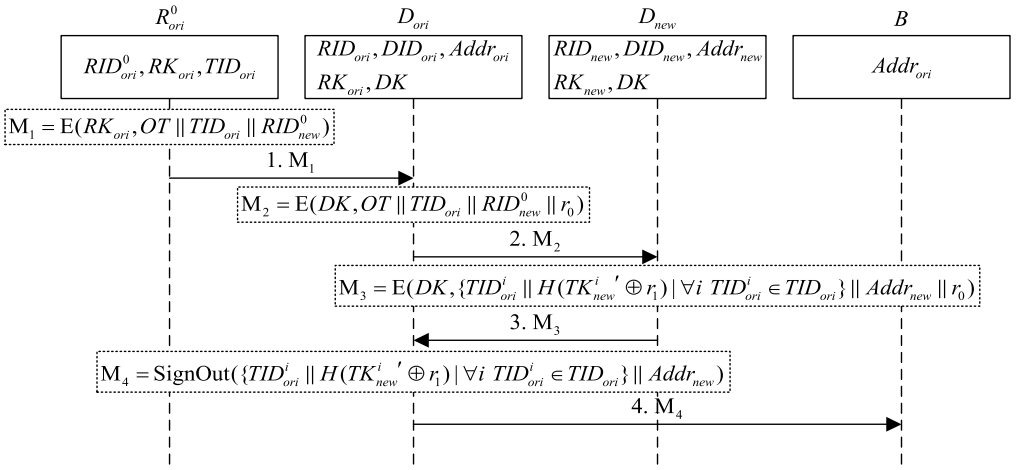


FIGURE 5. On-chain ownership outgoing.

which are then encrypted as M_5 by using RK_{ori}, the key shared by D_{ori} and R^0_{ori}. M_5 is then transmitted to R^0_{ori}.

After R^0_{ori} decrypts M_5 and authenticates its source using TID^i_{ori}, according to the group TID^i_{ori} belongs to, it outgoings M_5 and distributes the message to all auxiliary readers according to the number of tags each reader can afford to read. As shown in Fig. 6, after the j-th auxiliary reader (R^j_{ori}) receives the message, it decrypts the message into M_7 by using MK^j_{ori}, the key shared by R^j_{ori} and R^0_{ori}. A multicast message is then transmitted to the tags of the affiliated groupR^j_{ori}.

As illustrated in Fig. 6, after T^i_{ori}, the tag in the group governed by R^j_{ori}, receives the multicast message, decrypts M_7 by using TK^i_{ori}, the key shared by T^i_{ori} and D_{ori}, and authenticates the source of the message by verifying TID^i_{ori}, it generates a random value r^i_3. TK^i_{ori} is then used to encrypt r_2, r^i_3, and TID^i_{ori} into M^i_8, which is returned to R^j_{ori}, forwarded to R^0_{ori}, and finally returned to D_{ori}, thus completing the ownership outgoing. The actual product is thus transferred to the new owner (*new*), who then performs the ownership incoming.

B. OWNERSHIP INCOMING

After the new owner (*new*) receives the product (Fig. 7), the grouping proof of the product is generated before its tag is transferred to prevent subsequent disputes. Because *new* may have a different authority from that of the original owner (*ori*), the main reader of *new* (R^0_{new}) must first acquire authorization from D_{ori} through the backend server D_{new}. RK_{new}, the key shared by R^0_{new} and D_{new}, is used to encrypt the tag ID set of the received product TID_{ori} and the blockchain address of *new* ($Addr_{new}$) into M_1, which is then transmitted to D_{new}.

After D_{new} decrypts M_1, confirms the consistency between TID_{ori} and the tag ID of the transferred product TID_{ori}, and authenticates the source of the message, it generates a random number r_4. DK is then used to encrypt TID_{ori}, $Addr_{new}$, and r_4 into request message M_2, which is transmitted to D_{ori}.

After D_{ori} receives M_2 and confirms that TID_{ori} and $Addr_{new}$ are consistent with the transfer information, it generates the message M^j_{check} for all tags to return the grouping proof. DK is then used to encrypt the messages M^j_{check} from all the tag groups into M_3 for D_{new}, which then transmits M_4 to R^0_{new}.

After R^0_{new} decrypts M_4 and acquires the message required by each auxiliary reader. As Fig. 7, it defines M^j_{check} as M^j_5 to distribute to R^j_{new}, its j-th auxiliary reader R^j_{new} then employs the shared key MK^j_{new} to encrypt the message into M_6 for transmission to its group tags.

The tag T^i_{ori} decrypts M_6, authenticates its source, and verifies that its own ID TID^i_{ori} is correct and that the random number r^i_3 has been generated by itself through the outgoing process. The key TK^i_{ori} is applied to calculate MAC(TK^i_{ori}, $r_2 \| r^i_3$), and the message M^i_7 is returned to R^j_{new} and forwarded to R^0_{new}. R^0_{new} performs an XOR calculation on messages transmitted by all auxiliary readers and converts them into the grouping proof of the product, returning it to D_{new}.

After D_{new} receives the grouping proof, it retains the proof for use to solve any dispute that occurs in the transaction. The proof is signed using the private key PV_{new} and transmitted to D_{ori} through the message M_{10}.

After D_{ori} receives M_{10} and verifies the consistency of the grouping proof and the signature of *new* using PB_{new}, the public key of *new*, D_{ori} encrypts all the tag keys TK^i_{ori} into M_{11} with DK. M_{11} is then transmitted to D_{new} for key update.

After D_{new} receives M_{11}, it starts performing the on-chain and off-chain ownership transfer. As depicted in Fig. 8, D_{new} applies the updated key $TK^i_{new}{}'$ from the outgoing process and the random numbers generated by all the actors in the communication to update the key for each tag T^i_{ori}. The original TK^i_{new} is then encrypted into M^i_{renew}, which

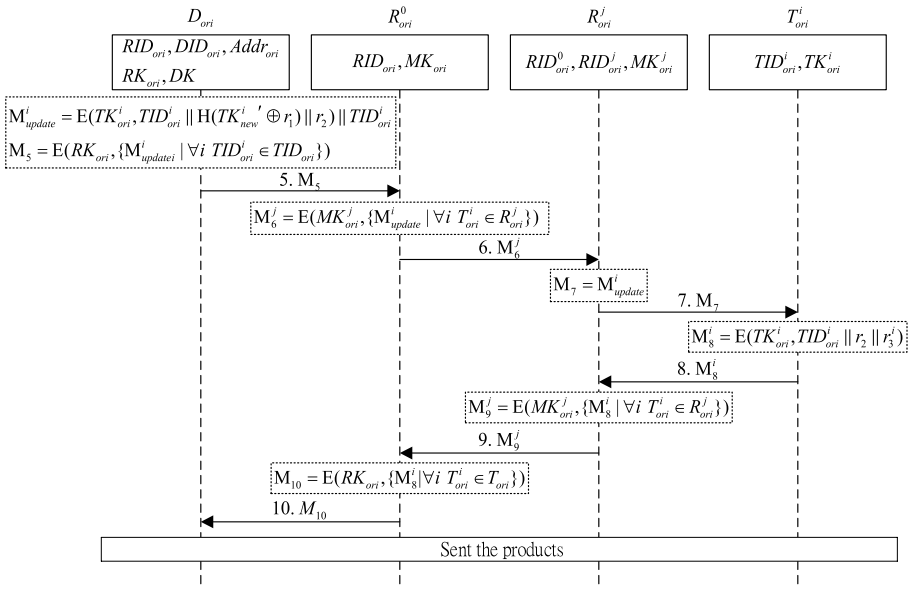


FIGURE 6. Off-chain ownership outgoing.

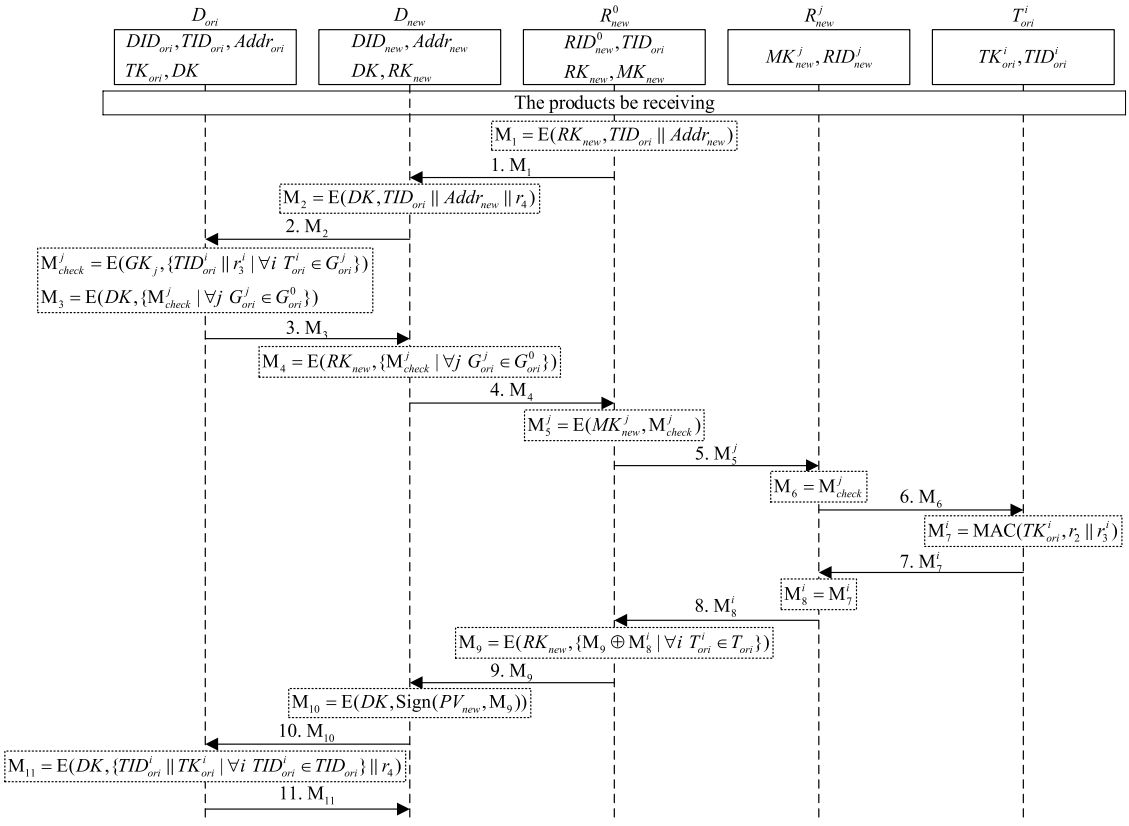


FIGURE 7. Generating the grouping proof of the product.

is further encrypted into M_{12} using RK_{new} and transmitted to R^0_{new}.

After R^0_{new} decrypts M_{12} and authenticates its source by using TID^i_{ori}, it acquires and distributes the message required for each auxiliary reader. As shown in Fig. 8, after R^j_{new} receives the message, it applies MK^j_{new} to decrypt the message into M^i_{14} for transmission to one of the tags under its administration (T^i_{ori}).

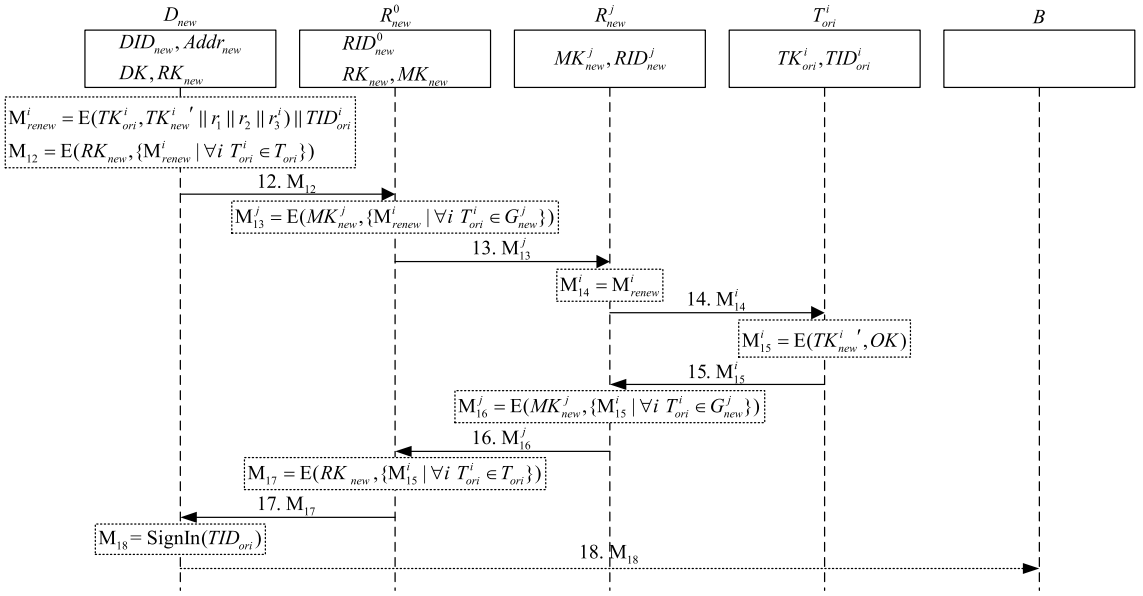


FIGURE 8. On-chain and off-chain ownership transfer.

After T^i_{ori} receives M^i_{14}, it recalculates the key hash, authenticates the message source, and confirms that the key hash and the random numbers r_2 and r^i_3 are consistent with M^j_{14}. The key in M^j_{14} is then updated as $TK^{i\,\prime}_{new}$, and its key hash and random numbers r_2 and r^i_3 are emptied. Subsequently, the message is encrypted into the completion message M^i_{15} by using a new tag key, returned to R^j_{new}, forwarded to R^0_{new}, and finally returned to D_{new}.

After D_{new} receives M_{17} from all the tags, it employs the incoming function SignIn() in the product management contract and the blockchain address $Addr_{new}$ to initiate the incoming process. The ownership of TID_{ori} is requested. After the product management contract confirms that $Addr_{new}$ belongs to the new owner, it generates a new ownership record, thus completing the on-chain and off-chain ownership incoming.

In the proposed protocol, all tags can be transferred with only one protocol execution step regardless of the total number of tags. The number of required auxiliary readers is related to the number of leaf group nodes. When all the tags to be transferred belong to the same leaf group node, only one auxiliary reader is required to complete their ownership transfer. Conversely, when the tags to be transferred belong to z different leaf group nodes, z auxiliary readers are required to complete their ownership transfer. When the number of auxiliary readers exceeds the upper limit of the main reader's simultaneous reading capacity (r), an additional auxiliary reader must be implemented to assist in the message transmission. Accordingly, the number of auxiliary readers is calculated as $z + \lceil z * r^{-1} \rceil + \lceil z * r^{-2} \rceil + \ldots + 1$.

As depicted in Fig. 3 (b), the original owner (*ori*) intends to transfer six tags (T^1_{ori}–T^6_{ori}) to the new owner (*new*). The

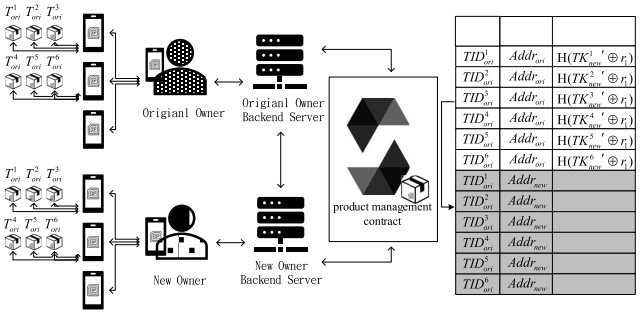


FIGURE 9. Transferring the ownership of two tag groups to the new owner simultaneously.

main reader of *ori* (R^0_{ori}) sends a transfer request message embedded with the tag IDs (TID^1_{ori}–TID^6_{ori}) to the server D_{ori}, which then requests the server of the new owner *new* (D_{new}) to update the message. After D_{new} receives the request, it generates six new tag key hash values ($H\left(TK^{1\,\prime}_{new} \oplus r_1\right)$, $H\left(TK^{2\,\prime}_{new} \oplus r_1\right)$, $\ldots H\left(TK^{6\,\prime}_{new} \oplus r_1\right)$). These hash values are integrated with the tag IDs (TID^1_{ori}–TID^6_{ori}) and the recipient blockchain address ($Addr_{new}$) and sent to D_{ori}. The product management contract in the D_{ori} blockchain then records the updated hash values, as listed in the table on the right of Fig. 9 After D_{ori} receives the random number r_2 generated by the contract, it distributes the hash values for updating the keys ($TID^1_{ori} \| H\left(TK^{1\,\prime}_{new} \oplus r_1\right)$, $TID^2_{ori} \| H\left(TK^{2\,\prime}_{new} \oplus r_1\right)$, $\ldots$, $TID^6_{ori} \| H\left(TK^{6\,\prime}_{new} \oplus r_1\right)$) and random number r_2 to the auxiliary readers (R^1_{ori} and R^2_{ori}) through R^0_{ori}.

Auxiliary reader R^1_{ori} receives from R^0_{ori} the update messages required for the tags T^1_{ori}, T^2_{ori}, and T^3_{ori}, which belong

to the same tag group. The messages are then transmitted to T^1_{ori}, T^2_{ori}, and T^3_{ori}. Similarly, R^2_{ori} receives from R^0_{ori} the update messages required for the tags T^4_{ori}, T^5_{ori}, and T^6_{ori}, which belong to the same tag group. The messages are then transmitted to T^4_{ori}, T^5_{ori}, and T^6_{ori}. After the tag has the source of its ownership transfer message and ID verified, a confirmation message is sent from R^1_{ori} or R^2_{ori} to D_{ori}. After D_{ori} confirms to have received the confirmation messages from all the tags, the products are transferred to the new owner.

After the new owner (*new*) receives the products, the main reader R^0_{new} acquires the ownership transfer messages encrypted with keys GK^4_{ori} and GK^5_{ori} respectively from D_{ori} through D_{new}. Through the auxiliary reader R^1_{new}, multicast ownership transfer messages encrypted with GK^4_{ori} are transmitted to T^1_{ori}, T^2_{ori}, and T^3_{ori}; similarly, through R^2_{new}, multicast ownership transfer messages encrypted with GK^5_{ori} are transmitted to T^4_{ori}, T^5_{ori}, and T^6_{ori}. After the tags receive the messages, they generate different parts of their grouping proof and return their messages. The main reader R^0_{new} then runs an XOR calculation on the returned messages to assemble the grouping proof, which is then transmitted to D_{new}. The grouping proof is signed by D_{new} by using its private key PV_{new} and transmitted to D_{ori}. Subsequently, D_{new} requests the tag keys TK^1_{ori}–TK^6_{ori} of the transferred products from D_{ori}.

After D_{ori} receives the message and confirms the authenticity of the grouping proof by verifying the signature with the new owner's public key PB_{new}, it transmits TK^1_{ori}–TK^6_{ori} to D_{new}. After D_{new} receives the tag keys, it generates a message with new tag keys $TK^{1\prime}_{new}$–$TK^{6\prime}_{new}$ and transmits it to R^1_{ori} or R^2_{ori} through R^0_{new}. The message is then distributed to T^1_{ori}, T^2_{ori}, and T^3_{ori} through R^1_{ori} and to T^4_{ori}, T^5_{ori}, and T^6_{ori} through R^2_{ori}. After each tag (e.g., T^1_{ori}) authenticates the source of the ownership transfer message and confirms the authenticity of the key (e.g., $TK^{1\prime}_{new}$) by using the hash value (e.g., $\mathrm{H}\left(TK^{1\prime}_{new} \oplus r_1\right)$), the product tag is updated as $TK^{i\prime}_{new}$, and the reader returns confirmation messages from tags to D_{new}. D_{new} confirms the completion of the update of all the tags and transmits the incoming message embedded with the tag ID set TID_{ori} to the blockchain, prompting the product management contract to record the ownership transfer as shown in the grey area of Fig. 9, thus completing the on-chain and off-chain ownership transfer.

V. SECURITY ANALYSIS

A total of five communication paths are involved in the proposed protocol, namely backend server to blockchain, backend server to backend server, backend server to main reader, main reader to auxiliary reader, and auxiliary reader to tag. The communication among backend server to blockchain, and backend server to backend server. Their computation power is strong enough for encryption algorithms like Advanced Encryption Standard (AES) [52] and RSA [53]. Because the messages between back-end servers are encrypted with secure cryptographic systems, wired network security issues will not be discussed in this paper. Communications between backend server to main reader, usually comply with security standards IEEE802.11i [54]. Therefore, we leave them out of discussion in this paper. Herein, the security of the communication between off-chain mobile readers and tags in the proposed protocol and of the on-chain smart contracts is assessed.

A. OFF-CHAIN SECURITY

Secure ownership transfer relies on preventing attacks during the transfer process. Therefore, a security analysis was conducted on the commonplace threats to ownership transfer, including security problems such as secret leakage, replay attacks, denial-of-service (DoS) attacks, man-in-the-middle (MitM) attacks, counterfeit tags or readers, and window problems as well as problems associated with forward security.

1) PREVENTING SECRET LEAKAGE

The ideal protocol must prevent attackers from acquiring sensitive information. In the proposed protocol, each message when transmitted is protected through symmetric key encryption, and the shared keys are deployed through secure channels in the initial stage. This prevents attackers from acquiring keys to decrypt the messages.

2) PREVENTING REPLAY ATTACKS

Attackers can eavesdrop and preserve the information previously transmitted by a protocol. Therefore, the ideal protocol must ensure that replayed old messages cannot evade authentication. In the proposed protocol, random numbers are generated by the backend servers of the original and new owners, tags, and blockchains and encrypted together with messages for transmission. Thus, the messages at each transfer contain random number changes to ensure message freshness, thereby preventing attackers from replaying acquired messages to evade authentication.

3) PREVENTING ASYNCHRONOUS DoS ATTACKS

During the updating of keys, the ideal protocol must prevent attackers from blocking the update process, which renders the keys out of sync and prevents them from being saved. In the proposed protocol, all the messages are encrypted with shared keys. Therefore, only the situations in which messages are lost or blocked by attackers are discussed. During the outgoing process, the original owner only enters the key hashes provided by the new owner into the tags and product management contract. When the hash values in the tags are not yet updated, the original owner may resend the messages. During the incoming process, the new owner saves the two keys before and after the update. Thus, the key before the update can still be used to communicate with the tags in the event the messages are blocked.

4) PREVENTING COUNTERFEIT TAG OR READER ATTACKS

The ideal protocol must prevent attackers from counterfeiting readers or tags to steal ownership. In the proposed protocol, attackers attempting to counterfeit tags must acquire tag

keys and IDs, which have been allocated to the tags through secure channels. Moreover, the confidentiality of transmission is protected, preventing attackers from counterfeiting tags. Attackers attempting to counterfeit readers must acquire reader keys and IDs, which have similarly been deployed through secure channels. Because the confidentiality of transmission is protected, attackers are unable to counterfeit readers.

5) PREVENTING MitM ATTACKS

Because the proposed protocol prevents attackers from counterfeiting tags or readers and from evading authentication through replay attacks, the protocol can secure against MitM attacks.

6) PREVENTING WINDOW PROBLEMS

During ownership transfer, the ideal protocol must ensure that the original and new owners do not possess the ownership simultaneously. In the proposed protocol, after the original owner enters the updated hash value provided by the new owner, the original owner can no longer modify the hash value with the original key and must use the key provided by the new owner to do so.

7) FORWARD SECRECY

When the original owner transfers tag ownership, the new owner provides only the updated hash values, which are entered into the tags by the original owner, thus completing the tag transfer. After the new owner acquires the tag keys, the tag keys are updated with new keys. This prevents the original owner from learning about the updated keys and tracing the subsequent tag information, thus protecting the forward secrecy.

Table 2 presents a comparison between the proposed protocol and the protocols in other studies in terms of the defense against Secret Leakage (SL), replay attacks (RA), DoS attacks (DoS), MitM attacks (MitM), counterfeit attacks (IA), and window problems (WP) as well as on forward security (FS), grouping proofs (GP), group transfer (GT), partial tag ownership transfer (POT), traceability (TB), and the ability to complete ownership transfer with only one protocol execution (OTF). Here, O indicates that the protocol fully prevents a particular type of attack or fulfills a particular characteristic; indicates that the protocol partially prevents particular type of attack or fulfills a particular characteristic; and X indicates that the protocol is completely incapable of preventing a particular type of attack or does not fulfill a particular characteristic.

According to Table 2, the protocols by Zuo [19] and Tsai et al. [38] are incapable of preventing some attacks or do not fulfill some security characteristics. The protocol by Zuo [19] is incapable of preventing asynchronous DoS attacks. Attackers can intercept the XOR calculation of multiple messages of preceding ownership transfer and replace the information with new keys, thus resulting in inconsistency between tags and the keys saved by the new owner and preventing tags from being updated by the new owner [11]. Moreover, the grouping proof generated by Zuo's protocol [19] updates its key after ownership transfer. This prevents tag groups from generating a grouping proof identical to the one in the authentication server so that the consistency in the number of products transferred can be verified, inhibiting the solution of disputes on incomplete ownership transfer. Finally, group numbers are assigned to tags in advance in the protocol by Zuo [19]. During the transfer process, all the tags within a group must be simultaneously transferred, and transferring only a part of the tags is impossible. The protocol by Tsai et al. [44] effectively prevents all the listed attacks. However, because it lacks a mechanism to trace manufacturers and does not record the history of ownership transfer, it is incapable of tracing the source manufacturers of products. Furthermore, when tags are divided into different groups, the protocol by Tsai et al. [44] must be executed multiple times to complete the ownership transfer, thereby lowering the transfer efficacy (see Section 5 for further details). By contrast, the protocol proposed in this study both effectively prevents most of the known ownership transfer attacks and traces tag sources. Additionally, it is the only protocol capable of completing key updates in only one execution step.

TABLE 2. Security comparison.

	Zuo[19]	Tsai et al.[44]	Our Protocol
SL	O	O	O
RA	O	O	O
DoS	X	O	O
MitM	O	O	O
IA	O	O	O
WP	O	O	O
FS	O	O	O
GP	△	O	O
GT	O	O	O
POT	X	O	O
OTF	X	X	O
TB	X	X	O

Next, we apply GNY logic [55] to proof the off-chain security of our proposed protocol. The verification has four parts:

1. Defines the message transferred in the protocol. (Table 4)
2. Assumptions about the initial state. (Table 5)
3. Goals of the protocol. (Table 6)
4. The process of the proof. (Table 7)

Table 3 defines the symbols used in the GNY logic proof. For the logic equation numbers used in Table 7, such as T1 and P1, please refer to GNY logic[55]. If initial assumptions are required, the term "IA" is used.

B. ON-CHAIN SECURITY

The security of the smart contracts in the proposed protocol were verified using Oyente, a smart contract security testing tool. Oyente reads smart protocols and detects several types of

TABLE 3. Definition of symbols used in the proof.

Symbol	Definition
D_{ori}	Original owner's server
D_{new}	New owner's server
R^0_{ori}	Original owner's main reader
R^0_{new}	New owner's main reader
r_1, r_2, r_3^i	Random number
GK_j	Group key
DK	Key shared by the backend servers of the original and new owners
MK_e^m	Key shared between the main mobile reader and the m-th auxiliary reader of owner, e
TK_e^i	Key shared by the product tag T_i and backend server owned by owner, e
RK_e	Key shared between the main mobile reader and backend server of owner, e
$\{X\}_K$,$\{X\}_K^{-1}$	Symmetric key K is used to encrypt or decrypt message X
$\{X\}_{+K}$,$\{X\}_{-K}$	Public key ($+K$) or private key ($-K$) is used to encrypt or decrypt message X
$P \lhd Msg1$	P receives message $Msg1$
$P \ni Msg1$	P owns message $Msg1$
$P \mid\equiv Q \mid \sim X$	P believes that Q transmits X
$P \mid\equiv \#(Msg1)$	P believes that message $Msg1$ is the first message it receives
$P \mid\equiv \emptyset(Msg1)$	P believes that message $Msg1$ is identifiable
$P \mid\equiv P \overset{s}{\leftrightarrow} Q$	P believes that key s is shared by P and Q
$P \mid\equiv P \overset{+s}{\leftrightarrow} Q$	P believes that key s is an open key of Q

TABLE 4. Protocol messages.

Ownership Outgoing	
M_6^j	$R^j_{ori} \lhd * \left\{TID^i_{ori}, \{TID^i_{ori}, H(TK^{i}_{new}{}' \oplus r_1), r_2\}_{TK^i_{ori}}\right\}_{MK^j_{ori}}$
M_7	$T^i_{ori} \lhd * \{TID^i_{ori}, H(TK^{i}_{new}{}' \oplus r_1), r_2\}_{TK^i_{ori}}$
M_8^i	$R^j_{ori} \lhd * \{TID^i_{ori}, r_2, r_3^i\}_{TK^i_{ori}}$
M_9^i	$R^o_{ori} \lhd * \{TID^i_{ori}, r_2, r_3^i\}_{MK^j_{ori}}$
M_{10}	$D_{ori} \lhd * \{TID^i_{ori}, r_2, r_3^i\}_{RK_{ori}}$
Ownership incoming	
M_5^j	$R^j_{new} \lhd * \left\{\{TID^i_{ori}, r_3^i\}_{GK_j}\right\}_{MK^j_{new}}$
M_6	$T^i_{ori} \lhd * \{TID^i_{ori}, r_3^i\}_{GK_j}$
M_7^i	$R^j_{new} \lhd * MAC\{r_2, r_3^i\}_{TK^i_{ori}}$
M_8^i	$R^o_{new} \lhd * MAC\{r_2, r_3^i\}_{TK^i_{ori}}$
M_9	$D_{new} \lhd * \left\{MAC\{r_2, r_3^i\}_{TK^i_{ori}}\right\}_{RK_{new}}$
M_{10}	$D_{ori} \lhd * \left\{Sign\left\{MAC\{r_2, r_3^i\}_{TK^i_{ori}}\right\}_{PV_{new}}\right\}_{DK}$
M_{11}	$D_{new} \lhd * \{TK^{i}_{new}{}', r_1, r_2, r_3^i, TID^i_{ori}\}_{TK^i_{ori}}$
M_{13}^j	$R^j_{new} \lhd * \left\{\{TK^{i}_{new}{}', r_1, r_2, r_3^i, TID^i_{ori}\}_{TK^i_{ori}}\right\}_{MK^j_{new}}$
M_{14}^i	$T^i_{ori} \lhd * \{TK^{i}_{new}{}', r_1, r_2, r_3^i, TID^i_{ori}\}_{TK^i_{ori}}$
M_{15}^i	$R^j_{new} \lhd * \{OK\}_{TK^i_{new}}$
M_{16}^j	$R^o_{new} \lhd * \{\{OK\}_{TK^i_{new}}\}_{MK^j_{new}}$

contract errors that may be vulnerable to attacks. Table 8 and Table 9 list the Oyente test results for the system management contract and product management contracts, respectively.

TABLE 5. Initial assumptions.

Back-end Server D_{ori}
$D_{ori} \ni RID_{ori}, DID_{ori}, Addr_{ori}, RK_{ori}, DK, TID_{ori}, TK_{ori}$
$D_{ori} \ni r_2$
$D_{ori} \mid\equiv \#(r_2)$
$D_{ori} \mid\equiv \emptyset(TID_{ori})$
$D_{ori} \mid\equiv D_{ori} \overset{DK}{\leftrightarrow} D_{new}$
$D_{ori} \mid\equiv D_{ori} \overset{TK_{ori}}{\longleftrightarrow} T^i_{ori}$
$D_{ori} \mid\equiv D_{new} \rightarrow D_{new} \mid\equiv *$
Back-end Server D_{new}
$D_{new} \ni RID_{new}, DID_{new}, Addr_{new}, RK_{new}, DK$
$D_{new} \ni r_1$
$D_{new} \mid\equiv \#(r_1)$
$D_{new} \mid\equiv D_{new} \overset{DK}{\leftrightarrow} D_{ori}$
$D_{new} \mid\equiv D_{ori} \rightarrow D_{ori} \mid\equiv *$
Reader R^0_{ori}
$R^o_{ori} \ni RID^o_{ori}, RK_{ori}, TID_{ori}, MK_{ori}$
$R^o_{ori} \mid\equiv R^o_{ori} \overset{RK_{ori}}{\longleftrightarrow} D_{ori}$
$R^o_{ori} \mid\equiv D_{ori} \rightarrow D_{ori} \mid\equiv *$
Reader R^0_{new}
$R^o_{new} \ni RID^o_{new}, RK_{new}, TID_{ori}, MK_{new}$
$R^o_{new} \mid\equiv R^o_{new} \overset{RK_{new}}{\longleftrightarrow} D_{new}$
$R^o_{new} \mid\equiv D_{new} \rightarrow D_{new} \mid\equiv *$
Reader R^j_{ori}
$R^j_{ori} \ni RID^o_{ori}, RID^j_{ori}, MK^j_{ori}$
$R^j_{ori} \mid\equiv R^j_{ori} \overset{MK^j_{ori}}{\longleftrightarrow} R^o_{ori}$
Reader RID^j_{new}
$R^j_{new} \ni RID^j_{new}, MK^j_{new}$
$R^j_{new} \mid\equiv R^j_{new} \overset{MK^j_{new}}{\longleftrightarrow} R^o_{new}$
Tags T^i_{ori}
$T^i_{ori} \ni TID^i_{ori}, TK^i_{ori}$
$T^i_{ori} \ni r_3^i$
$T^i_{ori} \mid\equiv \#(r_3^i)$
$T^i_{ori} \mid\equiv T^i_{ori} \overset{TK^i_{ori}}{\longleftrightarrow} D_{ori}$

The results confirm that both contracts are effectively protected against all common attacks.

VI. PERFORMANCE EVALUATION

Herein, the efficacy of the proposed group ownership transfer protocol incorporating blockchains is presented. The required on-chain calculation resources and the volumes of messages and calculations required for off-chain RFID ownership transfer were analyzed.

A. SMART CONTRACT PERFORMANCE EVALUATION

Geth was applied to provisions of three Ethereum nodes, namely the administrator, the manufacturer and the logistics company. Ethereum wallet was employed to calculate the processing fees required for the smart contracts when ownership is registered in Ethereum (Table 10).

1. 522,796 and 543,955 gas are required to deploy the system management contracts and product management contracts, respectively.

TABLE 6. Goals of the proposed protocol.

Goals:	
$D_{ori} \mid\equiv R^o_{ori}$ $\mid \sim \#\left(\left\{TID^i_{ori}, r_2, r^i_3\right\}_{RK_{ori}}\right)$ $D_{ori} \mid\equiv R^o_{ori} \mid \sim \emptyset\left(TID^i_{ori}\right)$ $D_{new} \mid\equiv T^i_{ori}$ $\mid \sim \#\left(\left\{TK^{i}_{new}{}', r_1, r_2, r^i_3, TID^i_{ori}\right\}_{TK^i_{ori}}\right)$ $D_{new} \mid\equiv \emptyset\left(TID^i_{ori}\right)$ $T^i_{ori} \mid\equiv R^j_{ori} \mid \sim \emptyset\left(TID^i_{ori}\right)$ $T^i_{ori} \mid\equiv D_{ori} \mid \sim \#\left(\left\{TID^i_{ori}, r^i_3\right\}_{GK_j}\right)$ $T^i_{ori} \mid\equiv R^j_{new} \mid \sim \emptyset\left(H\left(TK^{i}_{new}{}' \oplus r_1\right)\right)$ $T^i_{ori} \mid\equiv D_{new}$ $\mid \sim \#\left(\left\{TK^{i}_{new}{}', r_1, r_2, r^i_3, TID^i_{ori}\right\}_{TK^i_{ori}}\right)$ $D_{new} \mid\equiv D_{new} \overset{TK^{i}_{new}{}'}{\longleftrightarrow} T^i_{ori}$ $T^i_{ori} \mid\equiv D_{new} \overset{TK^{i}_{new}{}'}{\longleftrightarrow} T^i_{ori}$	1. Mutual authentication between group tags and the backend server 2. Ensure the effectiveness of the message; ensure that it is not a maliciously replayed message 3. Update the share key $TK^{i}_{new}{}'$ and random number r_1, r_2, r^i_3 of the group tag to those shared by the server of the new owner

2. Any manufacturer intending to register its products in the blockchain must apply for administrator review to earn authorized access to the product management contract. Each manufacturer, administrator must pay 125,922 gas to register a manufacturer in system management contracts. Besides, the manufacturer needs to pay 128,749 gas for confirmation.
3. Each manufacturer must submit the product's ID to the product management contracts and pay 113,362 gas to acquire initial ownership.
4. When a product is being transferred, its original owner must deliver an ownership outgoing application from the blockchain, and the new owner must then file an ownership incoming request. The requests are verified by the product management contract and the ownership transfer is completed. The original owner must pay 51,389 gas for the process, and the new owner must pay 19,741 gas to acquire the ownership of the product.

Fig. 10 illustrates the average processing fees required according to the number of times a product is transferred. When each manufacturer holds 10,000 products and transfers them 16 times, the average transfer cost is 78,223 gas. Because transferring cost shares the smart contract deployment cost, a contract is considerably more cost-efficient while increasing the number of transformations.

B. RFID PROTOCOL MESSAGE AND COMPUTATIONAL LOAD ANALYSIS

The message and computational loads required to transfer n tags were compared between the proposed protocol and the protocols by Zuo [19] and Tsai et al. [38]. In the frontend supply chain, manufacturers and wholesalers may simultaneously transfer a number of products exceeding the upper reading limit of a reader per operation, inhibiting transfer efficacy. Therefore, main readers are set up in the proposed protocol to distribute tag groups to different auxiliary readers, enabling auxiliary readers to read group tags simultaneously.

TABLE 7. Proof process.

Ownership Outgoing	
M^j_6:$R^j_{ori} \lhd* \left\{TID^i_{ori}, \left\{TID^i_{ori}, H\left(TK^{i}_{new}{}' \oplus r_1\right), r_2\right\}_{TK^i_{ori}}\right\}_{MK^j_{ori}}$ $R^j_{ori} \lhd \left\{TID^i_{ori}, \left\{TID^i_{ori}, H\left(TK^{i}_{new}{}' \oplus r_1\right), r_2\right\}_{TK^i_{ori}}\right\}_{MK^j_{ori}}$ /*T1*/ $R^j_{ori} \ni \left\{TID^i_{ori}, \left\{TID^i_{ori}, H\left(TK^{i}_{new}{}' \oplus r_1\right), r_2\right\}_{TK^i_{ori}}\right\}_{MK^j_{ori}}$ /*P1*/ $R^j_{ori} \mid\equiv R^0_{ori} \mid \sim \emptyset\left(\left\{TID^i_{ori}, \left\{TID^i_{ori}, H\left(TK^{i}_{new}{}' \oplus r_1\right), r_2\right\}_{TK^i_{ori}}\right\}_{MK^j_{ori}}\right)$ /*IA, I1, R2*/	After the j-th auxiliary reader R^j_{ori} receives the message, it uses the key MK^j_{ori} shared with main reader R^0_{ori} to decrypt the message, and then relay the message to T^i_{ori}.
M_7:$T^i_{ori} \lhd* \left\{TID^i_{ori}, H\left(TK^{i}_{new}{}' \oplus r_1\right), r_2\right\}_{TK^i_{ori}}$ $T^i_{ori} \lhd \left\{TID^i_{ori}, H\left(TK^{i}_{new}{}' \oplus r_1\right), r_2\right\}_{TK^i_{ori}}$ /*T1*/ $T^i_{ori} \ni \left\{TID^i_{ori}, H\left(TK^{i}_{new}{}' \oplus r_1\right), r_2\right\}_{TK^i_{ori}}$ /*P1*/ $T^i_{ori} \mid\equiv R^j_{ori} \mid \sim \emptyset\left(\left\{TID^i_{ori}, H\left(TK^{i}_{new}{}' \oplus r_1\right), r_2\right\}_{TK^i_{ori}}\right)$ /*IA, I1, R2*/ $T^i_{ori} \mid\equiv R^j_{ori} \mid \sim \emptyset\left(TID^i_{ori}\right)$ /*IA, I1*/	The tag T^i_{ori} in the group governed by R^j_{ori}, T^i_{ori} uses the key TK^i_{ori} shared with D_{ori} to decrypt the message M_7 .To recognize message come from D_{ori}.
M^i_8:$R^j_{ori} \lhd* \left\{TID^i_{ori}, r_2, r^i_3\right\}_{TK^i_{ori}}$ $R^j_{ori} \lhd \left\{TID^i_{ori}, r_2, r^i_3\right\}_{TK^i_{ori}}$ /*T1*/ $R^j_{ori} \ni \left\{TID^i_{ori}, r_2, r^i_3\right\}_{TK^i_{ori}}$ /*P1*/ $R^j_{ori} \mid\equiv T^i_{ori} \mid \sim \emptyset\left(TID^i_{ori}\right)$ /*IA*/	R^j_{ori} identify TID^i_{ori} is governed by itself, and then relay the message to R^o_{ori} .
M^i_9: $R^o_{ori} \lhd* \left\{TID^i_{ori}, r_2, r^i_3\right\}_{MK^j_{ori}}$ $R^o_{ori} \lhd \left\{TID^i_{ori}, r_2, r^i_3\right\}_{MK^j_{ori}}$ /*T1*/ $R^o_{ori} \ni \left\{TID^i_{ori}, r_2, r^i_3\right\}_{MK^j_{ori}}$ /*P1*/ $R^o_{ori} \mid\equiv R^j_{ori} \mid \sim \emptyset\left(\left\{TID^i_{ori}, r_2, r^i_3\right\}_{MK^j_{ori}}\right)$ /*IA, I1, R2*/	R^o_{ori} uses the key MK^j_{ori} shared with R^j_{ori} to decrypt the message. To ensure the message come from the reader R^j_{ori}.
M_{10}: $D_{ori} \lhd* \left\{TID^i_{ori}, r_2, r^i_3\right\}_{RK_{ori}}$ $D_{ori} \lhd \left\{TID^i_{ori}, r_2, r^i_3\right\}_{RK_{ori}}$ /*T1*/ $D_{ori} \ni \left\{TID^i_{ori}, r_2, r^i_3\right\}_{RK_{ori}}$ /*P1*/ $D_{ori} \mid\equiv R^o_{ori} \mid \sim \emptyset\left(\left\{TID^i_{ori}, r_2, r^i_3\right\}_{RK_{ori}}\right)$ /*IA, I1, R2*/ $D_{ori} \mid\equiv R^o_{ori} \mid \sim \emptyset\left(TID^i_{ori}\right)$ /*IA, I1, R2*/ $D_{ori} \mid\equiv R^o_{ori} \mid \sim \emptyset(r_2)$ /*IA*/ $D_{ori} \mid\equiv R^o_{ori} \mid \sim \#\left(\left\{TID^i_{ori}, r_2, r^i_3\right\}_{RK_{ori}}\right)$ /*IA, I1, F2*/	D_{ori} uses the key RK_{ori} shared with R^o_{ori} to decrypt the message. To ensure the message come from R^o_{ori} . It identifies the freshness of random number r_2, to ensure the message is fresh.
ownership incoming:	
M^j_5: $R^j_{new} \lhd* \left\{\left\{TID^i_{ori}, r^i_3\right\}_{GK_j}\right\}_{MK^j_{new}}$ $R^j_{new} \lhd \left\{\left\{TID^i_{ori}, r^i_3\right\}_{GK_j}\right\}_{MK^j_{new}}$ /*T1*/ $R^j_{new} \ni \left\{\left\{TID^i_{ori}, r^i_3\right\}_{GK_j}\right\}_{MK^j_{new}}$ /*P1*/ $R^j_{new} \mid\equiv R^o_{new} \mid \sim \emptyset\left(\left\{\left\{TID^i_{ori}, r^i_3\right\}_{GK_j}\right\}_{MK^j_{new}}\right)$ /*IA, I1, R2*/	R^j_{new} uses the key MK^j_{new} shared with main reader R^0_{new} to decrypt the message, and then directly transmits the message to T^i_{new}.
M_6:$T^i_{ori} \lhd* \left\{TID^i_{ori}, r^i_3\right\}_{GK_j}$ $T^i_{ori} \lhd \left\{TID^i_{ori}, r^i_3\right\}_{GK_j}$ /*T1*/	T^i_{ori} uses the key GK_j to decrypt the message M_6, to ensure that message

TABLE 7. *(Continued.)* **Proof process.**

$T_{ori}^{i} \ni \{TID_{ori}^{i}, r_3^i\}_{GK_j}$ /*P1*/ $T_{ori}^{i} \mid\equiv R_{ori}^{j} \mid \sim\emptyset\left(\{TID_{ori}^{i}, r_3^i\}_{GK_j}\right)$/*IA, I1, R2*/ $T_{ori}^{i} \mid\equiv R_{ori}^{j} \mid \sim\emptyset(TID_{ori}^{i})$ /*IA, I1*/ $T_{ori}^{i} \mid\equiv \emptyset(r_3^i)$ /*IA*/ $T_{ori}^{i} \mid\equiv \#\left(\{TID_{ori}^{i}, r_3^i\}_{GK_j}\right)$/*IA, I1, F2*/	come from D_{ori}. T_{ori}^{i} verifies the random number r_3^i has been generated by itself in this tractions. To ensure that message is fresh.
M_7^i:$R_{new}^{j} \lhd* MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}$ $R_{new}^{j} \lhd MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}$ /*T1*/ $R_{new}^{j} \ni MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}$ /*P1*/	R_{new}^{j} relay the message to R_{ori}^{o} .
M_8^i:$R_{new}^{o} \lhd* MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}$ $R_{new}^{o} \lhd MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}$ /*T1*/ $R_{new}^{o} \ni MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}$ /*P1*/	R_{new}^{0} relay the message to D_{new}.
M_9:$D_{new} \lhd* \left\{MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}\right\}_{RK_{new}}$ $D_{new} \lhd \left\{MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}\right\}_{RK_{new}}$ /*T1*/ $D_{new} \ni \left\{MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}\right\}_{RK_{new}}$ /*P1*/ $D_{new} \mid\equiv R_{new}^{o} \mid \sim\emptyset\left(\left\{MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}\right\}_{RK_{new}}\right)$ /*IA, I1, R2*/	D_{ori} uses the key RK_{new} shared with R_{new}^{0} to decrypt the message. To ensure the message come from R_{new}^{0}.
M_{10}:$D_{ori} \lhd* \left\{Sign\left\{MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}\right\}_{PV_{new}}\right\}_{DK}$ $D_{ori} \lhd \left\{Sign\left\{MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}\right\}_{PV_{new}}\right\}_{DK}$ $D_{ori} \ni \left\{Sign\left\{MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}\right\}_{PV_{new}}\right\}_{DK}$ $D_{ori} \mid\equiv \emptyset(r_2)$ /*IA, I1*/ $D_{ori} \mid\equiv \emptyset\left(Sign\left\{MAC\{r_2, r_3^i\}_{TK_{ori}^{i}}\right\}_{PV_{new}}\right)$ /*IA, I1, T6*/	D_{ori} uses the key DK shared with D_{new} to decrypt the message. To ensure the message come from D_{new} . It identifies the freshness of random number r_2, to ensure the message is fresh.
M_{11}:$D_{new} \lhd* \{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}$ $D_{new} \lhd \{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}$ /*T1*/ $D_{new} \ni \{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}$ /*P1*/ $D_{new} \mid\equiv D_{ori} \mid \sim\emptyset\left(\{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}\right)$/*IA, I1, R2*/ $D_{new} \mid\equiv \emptyset(r_1, r_2, r_3^i)$ /*IA,I1*/ $D_{new} \mid\equiv \emptyset(TID_{ori}^{i})$ /*IA, I1*/ $D_{new} \mid\equiv \#(r_1)$ /*IA*/ $D_{new} \mid\equiv T_{ori}^{i} \mid \sim\#\left(\{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}\right)$ /*IA, I1, F2*/ $D_{new} \mid\equiv D^{j} \overset{TK_{new}^{i}{}'}{\longleftrightarrow} T_{ori}^{i}$ /*J1*/	D_{new} uses the keyTK_{ori}^{i} shared with T_{ori}^{i} to decrypt the message. To ensure that message come from T_{ori}^{i} . D_{new} verifies the random number r_1 has been generated by itself in this tractions. To ensure that message is fresh. D_{new} updates the key $TK_{new}^{i}{}'$ for each tag T_{ori}^{i}.
M_{13}^{j}: $R_{new}^{j} \lhd* \left\{\{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}\right\}_{MK_{new}^{j}}$ $R_{new}^{j} \lhd \left\{\{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}\right\}_{MK_{new}^{j}}$ /*T1*/ $R_{new}^{j} \ni \left\{\{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}\right\}_{MK_{new}^{j}}$ /*P1*/ $R_{new}^{j} \mid\equiv R_{new}^{o} \mid \sim\emptyset\left(\left\{\{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}\right\}_{MK_{new}^{j}}\right)$ /*IA, I1, R2*/	R_{new}^{j} uses the key MK_{new}^{j} shared with main reader R_{new}^{0} to decrypt the message, and relay the message to T_{ori}^{i}.

Because the proposed protocol is the first secure ownership transfer protocol that reads group tags simultaneously

TABLE 7. *(Continued.)* **Proof process.**

M_{14}^{i}: $T_{ori}^{i} \lhd* \{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}$ $T_{ori}^{i} \lhd \{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}$ /*T1*/ $T_{ori}^{i} \ni \{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}$ /*P1*/ $T_{ori}^{i} \mid\equiv R_{new}^{j} \mid \sim\emptyset\left(\{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}\right)$/*IA, I1, R2*/ $T_{ori}^{i} \mid\equiv R_{new}^{j} \mid \sim\emptyset\left(H\left(TK_{new}^{i}{}' \oplus r_1\right)\right)$ /*IA, I1, R2*/ $T_{ori}^{i} \mid\equiv \emptyset(r_2, r_3^i)$ /*IA*/ $T_{ori}^{i} \mid\equiv \#\left(H\left(TK_{new}^{i}{}' \oplus r_1\right)\right)$ /*IA, I1, F2*/ $T_{ori}^{i} \mid\equiv D_{new} \mid \sim\#\left(\{TK_{new}^{i}{}', r_1, r_2, r_3^i, TID_{ori}^{i}\}_{TK_{ori}^{i}}\right)$ /*IA, I1, F2*/ $T_{ori}^{i} \mid\equiv D^{j} \overset{TK_{new}^{i}{}'}{\longleftrightarrow} T_{ori}^{i}$/*J1*/	T_{ori}^{i} uses the key GK_j to decrypt the message M_{14}^{i}, to ensure that message come from D_{new} . T_{ori}^{i} verifies the random number r_3^i has been generated by itself in this tractions. To ensure that message is fresh.T_{ori}^{i} updates the key $TK_{new}^{i}{}'$.

TABLE 8. **Oyente test results for the resistance of the system management contract against attacks.**

oyente version 0.2.7 – Commonwealth contract system. sol:System:	
EVM Code Coverage	99.9%
Integer Underflow	False
Integer Overflow	False
Parity Multisig Bug 2	False
Callstack Depth Attack Vulnerability	False
Transaction-Ordering Dependence (TOD)	False
Timestamp Dependency	False
Re-Entrancy Vulnerability	False

TABLE 9. **Oyente test results for the resistance of the product management contract against attacks.**

oyente version 0.2.7 – Commonwealth contract manager.sol:Manager:	
EVM Code Coverage	83.7%
Integer Underflow	False
Integer Overflow	False
Parity Multisig Bug 2	False
Callstack Depth Attack Vulnerability	False
Transaction-Ordering Dependence (TOD)	False
Timestamp Dependency	False
Re-Entrancy Vulnerability	False

through multiple readers, an overhead is applied to coordinate readers for reading multiple tag groups simultaneously in a secure manner. To fairly compare the efficacy of the proposed protocol with that of other group ownership transfer protocols, multiple messages parallelly transmitted by readers are defined as one message, and the parallel calculations conducted simultaneously in different readers are not repeated.

Currently, only the proposed protocol and the protocols by Zuo [19] and Tsai et al. [38] are capable of performing simultaneous group ownership transfer and grouping proof generation. These protocols were compared, and the results

TABLE 10. Processing fee for each contract interaction.

Behavior	Processing fee
Deploy the system management contract	522796 gas
Deploy the product management contract	543955 gas
Register manufacturer information	125922 gas
Authenticate manufacturer information	128749 gas
Acquire initial product ownership	113362 gas
Outgoing ownership	51389 gas
Incoming ownership	19741 gas

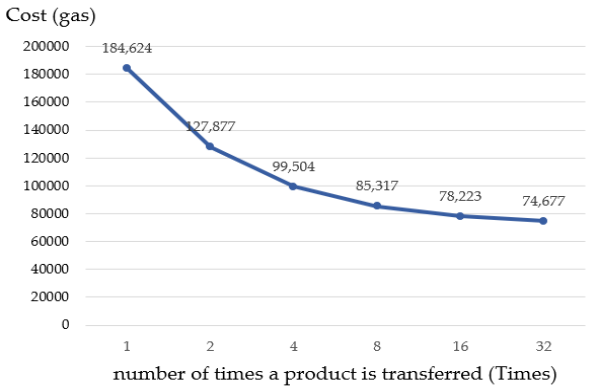


FIGURE 10. Average cost when a manufacturer transfers 10,000 products.

TABLE 11. Number of messages required to transfer *n* tags to the new owner.

	Number of messages
Zuo[19]	$6+8n$
Tsai et al.[44]	$(8+5k)\left\lceil\frac{n}{k}\right\rceil$
Our Protocol	$16+5k+5\left\lceil\frac{n}{k}\right\rceil$

verified that the proposed protocol is the most efficient for secure frontend logistics ownership transfer.

When a reader uses a k-ary group tag key tree to transfer n tags, $\lceil n/k \rceil$ groups must be transferred. Table 11 lists the number of messages required by the three mentioned protocols to perform ownership transfer. The off-chain outgoing and incoming stages require $6+k+2\left\lceil\frac{n}{k}\right\rceil$ and $10+4k+3\left\lceil\frac{n}{k}\right\rceil$ messages, respectively. Accordingly, completing a transfer process requires a total of $16+5k+5\left\lceil\frac{n}{k}\right\rceil$. In the protocol by Tsai et al. [44], outgoing readers distribute group keys to incoming readers in an out-of-band manner and do not integrate key distribution into the protocol. Consequently, tags belonging to different groups cannot be simultaneously read and must be read in separate protocol executions. Therefore, a relatively large number of messages are required for the simultaneous processing of tag groups in the proposed protocol of this study. Because the protocol by Zuo [49] does not support partial ownership transfer and updates an entire tag group at one time with a group key, fewer messages are required for ownership transfer using this protocol. However,

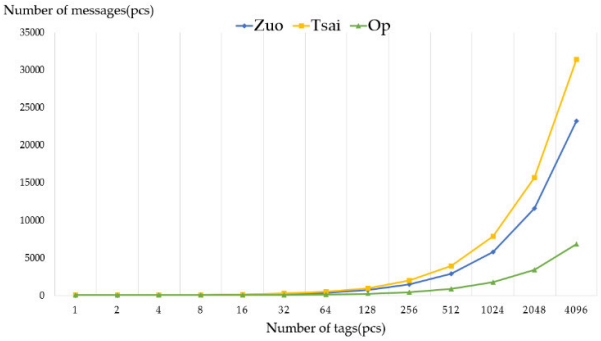


FIGURE 11. Computational load required for tag transfer by using a 3-ary key tree (k = 3).

generating a grouping proof in this protocol requires the sequential cascading of all tags; this requires a higher number of messages than does broadcasting messages to all tags, through which grouping proofs can be generated without following the tag sequence. Moreover, because the protocol proposed by Zuo [49] does not employ multiple readers to read tags simultaneously, it requires the highest number of messages to complete ownership transfer and generate grouping proofs.

Fig. 11 presents a comparison between the proposed protocol, the protocols by Zuo [49] and Tsai et al. [37] in terms of the number of messages required to complete ownership transfer. When a 3-ary key tree is used for the secure transmission of each tag group, with a maximum of only three tags, the present proposed protocol requires fewer messages than the other two previous protocols for simultaneously processing more than five tags; the difference is particularly pronounced when more than 128 tags must be processed. Thus, for the simultaneous transfer of ownership of a massive number of tags, the proposed protocol is the most efficient, with a reduced calculation time, thus making it most suitable for bulk cargo ownership transfer in the frontend supply chain environment.

Table 12 lists the computational loads required for each facility, where T_E indicates the time required for symmetric encryption and decryption; T_S refers to the time required for signature; T_H indicates the time required for calculating the hash value; and T_{RNG} is the time required for generating a random number and a key. Logic operations such as XOR are not discussed because they require much shorter calculation times than the mentioned variables. The protocol by Zuo [19] features owners as server with high calculation capabilities; in the protocol by Tsai et al. [44], couriers and recipients are designated as servers for analyzing calculation capabilities. Nevertheless, the protocol proposed in this study requires the lowest volume of calculation for tags and readers with low calculation capabilities.

For easy comparison of the efficacies of the three protocols, the computational loads for all the facilities in each protocol were summed (Table 13). Although the protocol proposed in this study requires the lowest computational

TABLE 12. Computational load required for each facility to transfer *n* tags.

Protocol	Facility		Computational load
Zuo[19]	Authentication server		$\lceil \frac{n}{k} \rceil T_D + \lceil \frac{n}{k} \rceil T_E + 2nT_{RNG}$
	Original owner	Backend server	$3\lceil \frac{n}{k} \rceil T_D + 2\lceil \frac{n}{k} \rceil T_E$
		Main reader	$\left(\lceil \frac{n}{k} \rceil + 2n\right) T_{RNG} + 4nT_H$
		Auxiliary reader	N/A
	New owner	Backend server	$\left(\lceil \frac{n}{k} \rceil + n\right) T_E + \left(2\lceil \frac{n}{k} \rceil + 2n\right) T_{RNG}$
		Main reader	N/A
		Auxiliary reader	N/A
	Tag		$3nT_{RNG}$
Tsai et al.[44]	Authentication server		$2nT_E + nT_{RNG}$
	Original owner	Backend server	$\lceil \frac{n}{k} \rceil T_S + \lceil \frac{n}{k} \rceil T_{RNG}$
		Main reader	$2\left(\lceil \frac{n}{k} \rceil + n\right) T_E + \lceil \frac{n}{k} \rceil T_H + \lceil \frac{n}{k} \rceil T_{RNG}$
		Auxiliary reader	$\left(3\lceil \frac{n}{k} \rceil + 2n\right) T_E + \lceil \frac{n}{k} \rceil T_{RNG} + 2\lceil \frac{n}{k} \rceil T_H$
	New owner	Backend server	$\lceil \frac{n}{k} \rceil T_S + \lceil \frac{n}{k} \rceil T_{RNG}$
		Main reader	N/A
		Auxiliary reader	N/A
	Tag		$\left(\lceil \frac{n}{k} \rceil + n\right) T_E + \left(\lceil \frac{n}{k} \rceil\right) T_{RNG} + \left(3\lceil \frac{n}{k} \rceil\right) T_H$
Our Protocol	Authentication server		N/A
	Original owner	Backend server	$\left(9 + n + \lceil \frac{n}{k} \rceil\right) T_E + T_{RNG}$
		Main reader	$\left(3 + 2\lceil \frac{n}{k} \rceil\right) T_E$
		Auxiliary reader	$2T_E$
Our Protocol	New owner	Backend server	$(11 + n)T_E + T_S + (2 + n)T_{RNG} + nT_H$
		Main reader	$\left(5 + 3\lceil \frac{n}{k} \rceil\right) T_E$
		Auxiliary reader	$4T_E$
	Tag		$(2 + 3k)T_E + T_H + T_{RNG}$

loads for facilities with low calculation capabilities, such as tags and readers, compared with the other two protocols, the required calculation in the proposed protocol is primarily concentrated in servers. Therefore, the comparison approach is relatively unfair to the proposed protocol. Nevertheless, the proposed protocol still requires considerably lower computational loads than the other two protocols.

To accurately and fairly compare the computational load of the three protocols, AES-128 was set as the general symmetric key encryption method. Each encryption or decryption requires 1,032 cycles [56]. Given that an RFID tag

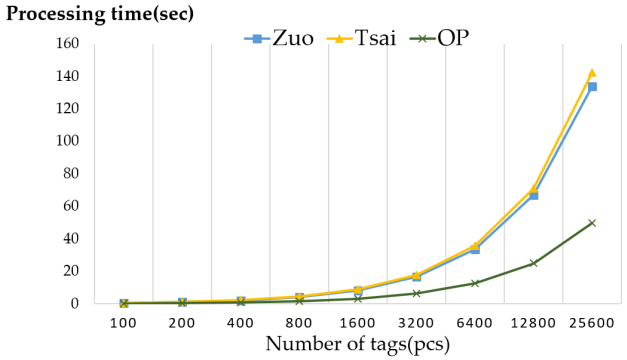


FIGURE 12. Computational load required for tag transfer by using a 3-ary key tree (k = 3).

TABLE 13. Total computational load required for transferring *n* tags.

Protocol	Total computational load
Zuo[19]	$\left(11\lceil \frac{n}{k} \rceil + n\right) T_E + \left(3\lceil \frac{n}{k} \rceil + 9n\right) T_{RNG} + 4nT_H$
Tsai et al.[44]	$\left(6\lceil \frac{n}{k} \rceil + 7n\right) T_E + 2\lceil \frac{n}{k} \rceil T_S + \left(5\lceil \frac{n}{k} \rceil + n\right) T_{RNG} + 6\lceil \frac{n}{k} \rceil T_H$
Our Protocol	$\left(34 + 6\lceil \frac{n}{k} \rceil + 2n + 3k\right) T_E + T_S + (4 + n)T_{RNG} + (1 + n)T_H$

can execute 3.55 million clock cycles per second [57], the clock cycle was calculated, followed by the time required to complete ownership transfer. As depicted in Fig. 12, although the comparison method is disadvantageous to the proposed protocol, the proposed protocol requires half the calculation time as that required by other protocols, and the difference increases with the increase in the number of tags. Accordingly, both the message and computational load comparisons indicate that the proposed protocol is the most suitable for transferring the ownership of a large number of products in frontend logistics.

VII. CONCLUSION

We have proposed a blockchain-based high-efficacy group ownership transfer protocol that suitable for bulk cargo transfer in a frontend supply chain environment. The advantage of this protocol is that product ownership transfer history recorded using the blockchain, which cannot be tampered with. This enables consumers to use the blockchain to trace the original manufacturers of all products and thereby prevent counterfeiting. Moreover, the protocol generates grouping proofs to prevent disputes regarding the comprehensiveness of ownership transfer.

Our agreement enables efficient transfer of the ownership of one or more RFID tags simultaneously. In processing more than five tags simultaneously, the proposed protocol requires fewer messages than the existing group ownership transfer protocols with grouping proofs; the difference is particularly pronounced when more than 128 tags are transferred. The experiment results revealed that the calculation time required by the proposed protocol was half of that required by other

protocols. Thus, for the simultaneous transfer of ownership of a massive number of tags, the proposed protocol is the most efficient.

We used GNY logic to verify the off-chain security of the proposed group ownership transfer protocol. We verify that participants can achieve mutual authentication between group tags and the backend server while ensuring that messages are not maliciously replayed, through GNY logic. Security analysis results have shown that the proposed protocol can prevent most RFID ownership transfer attacks, such as SL, replay, DoS, counterfeit tag or reader, and MitM attacks.

Furthermore, Oyente was employed to test the on-chain security of the smart contracts to ensure that these contracts are capable of resisting all commonplace attacks, thereby verifying that the proposed protocol is secure for both on-chain and off-chain transfers.

REFERENCES

[1] T. Sabanoglu. *Retail E-Commerce Sales Worldwide From 2014 to 2024*. Beeketing. (Accessed: May 10, 2021). [Online]. Available: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales

[2] K. Toyod, P. T. Mathiopoulo, I. Sasase, and T. Ohtsuk, "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.

[3] K. H. S. S. Koralalage, S. M. Reza, J. Miura, Y. Goto, and J. Cheng, "POP method: An approach to enhance the security and privacy of RFID systems used in product lifecycle with an anonymous ownership transferring mechanism," in *Proc. ACM Symp. Appl. Comput. (SAC)*, 2007, pp. 270–275.

[4] K. Osaka, T. Takagi, K. Yamazaki, and O. Takahashi, "An efficient and secure RFID security method with ownership transfer," in *Proc. Int. Conf. Comput. Intell. Secur.* Cham, Switzerland: Springer, Nov. 2006, pp. 147–176.

[5] H. Lei and T. Cao, "RFID protocol enabling ownership transfer to protect against traceability and DoS attacks," in *Proc. 1st Int. Symp. Data, Privacy, E-Commerce (ISDPE)*, Nov. 2007, pp. 508–510.

[6] C.-H. Wang and S. Chin, "A new RFID authentication protocol with ownership transfer in an insecure communication environment," in *Proc. 9th Int. Conf. Hybrid Intell. Syst.*, vol. 1, 2009, pp. 486–491.

[7] S. Fouladgar and H. Afifi, "An efficient delegation and transfer of ownership protocol for RFID tags," in *Proc. 1st Int. EURASIP Workshop RFID Technol.*, Vienna, Austria, vol. 66, 2007, pp. 68–93.

[8] S. Fouladgar and H. Afifi, "A simple privacy protecting scheme enabling delegation and ownership transfer for RFID tags," *J. Commun.*, vol. 2, no. 6, pp. 6–13, 2007.

[9] S. Fouladgar and H. Afifi, "A simple delegation scheme for RFID systems (SiDeS)," in *Proc. IEEE Int. Conf. RFID*, Dec. 2007, pp. 1–6.

[10] M. H. Yang and Y. H. Hu, "Protocol for ownership transfer across authorities: With the ability to assign transfer target," *Secur. Commun. Netw.*, vol. 5, no. 2, pp. 164–177, 2012.

[11] H. Jannati and A. Falahati, "Cryptanalysis and enhancement of a secure group ownership transfer protocol for RFID tags," in *Global Security, Safety and Sustainability & e-Democracy*. Cham, Switzerland: Springer, 2011, pp. 186–193.

[12] L. He, Y. Gan, and Y. Yin, "Secure group ownership transfer protocol with independence of old owner for RFID tags," *Comput. Model. New Technol.*, vol. 18, no. 12B, pp. 209–214, 2014.

[13] G. Kapoor, W. Zhou, and S. Piramuthu, "Multi-tag and multi-owner RFID ownership transfer in supply chains," *Decis. Support Syst.*, vol. 52, no. 1, pp. 258–270, Dec. 2011.

[14] N. Bagheri, S. F. Aghili, and M. Safkhani, "On the security of two ownership transfer protocols and their improvements," *Int. Arab J. Inf. Technol.*, vol. 15, no. 1, pp. 87–93, 2018.

[15] M. H. Yang, "Secure multiple group ownership transfer protocol for mobile RFID," *Electron. Commerce Res. Appl.*, vol. 11, no. 4, pp. 361–373, 2012.

[16] M.-L. Tsai and Y.-Y. Chen, "A novel group ownership proof and transfer scheme for B2B, B2C and C2C transactions," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 29, nos. 1–2, pp. 28–40, 2018.

[17] C.-C. Lee, C.-T. Li, C.-L. Cheng, Y.-M. Lai, and A. V. Vasilakos, "A novel group ownership delegate protocol for RFID systems," *Inf. Syst. Frontiers*, vol. 21, no. 5, pp. 1153–1166, Oct. 2019.

[18] F. Moazami and M. Safkhani, "SEOTP: A new secure and efficient ownership transfer protocol based on quadric residue and homomorphic encryption," *Wireless Netw.*, vol. 26, no. 7, pp. 5285–5306, Oct. 2020.

[19] Y. Zuo, "Changing hands together: A secure group ownership transfer protocol for RFID tags," in *Proc. 43rd Hawaii Int. Conf. Syst. Sci.*, 2010, pp. 1–10.

[20] K. Michael and L. McCathie, "The pros and cons of RFID in supply chain management," in *Proc. Int. Conf. Mobile Bus. (ICMB)*, 2005, pp. 623–629.

[21] R. Moraca and R. Hollinger. (2018). National Retail Security Survey. National Retail Federation. Accessed: Oct. 12, 2022. [Online]. Available: https://cdn.nrf.com/sites/default/files/2018-10/NRF-NRSS-Industry-Research-Survey-2018.pdf

[22] T. K. Agrawal, V. Kumar, R. Pal, L. Wang, and Y. Chen, "Blockchain-based framework for supply chain traceability: A case example of textile and clothing industry," *Comput. Ind. Eng.*, vol. 154, Apr. 2021, Art. no. 107130.

[23] (2013). *Object Name Service (ONS) GS1*. [Online]. Available: https://www.gs1.org/standards/epcis/epcis-ons/2-0-1

[24] R. Cole, M. Stevenson, and J. Aitken, "Blockchain technology: Implications for operations and supply chain management," *Supply Chain Manag., Int. J.*, vol. 24, no. 4, pp. 469–483, Jun. 2019.

[25] S. E. Chang and Y. Chen, "When blockchain meets supply chain: A systematic literature review on current development and potential applications," *IEEE Access*, vol. 8, pp. 62478–62494, 2020.

[26] A. Juels, "'Yoking-proofs' for RFID tags," in *Proc. IEEE Annu. Conf. Pervasive Comput. Commun. Workshops*, Sep. 2004, pp. 138–143.

[27] J. Saito and K. Sakurai, "Grouping proof for RFID tags," in *Proc. 19th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, vol. 2, Mar. 2005, pp. 621–624.

[28] S. Piramuthu, "On existence proofs for multiple RFID tags," in *Proc. ACS/IEEE Int. Conf. Pervasive Services*, Jun. 2006, pp. 317–320.

[29] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "Solving the simultaneous scanning problem anonymously: Clumping proofs for RFID tags," in *Proc. 3rd Int. Workshop Secur., Privacy Trust Pervasive Ubiquitous Comput. (SecPerU)*, Jul. 2007, pp. 55–60.

[30] P. Peris-Lopez, M. Safkhani, N. Bagheri, and M. Naderi, "RFID in eHealth: How to combat medication errors and strengthen patient safety," *J. Med. Biol. Eng*, vol. 33, pp. 363–372, Jan. 2013.

[31] L. Bolotnyy and G. Robins, "Generalized 'yoking-proofs' for a group of RFID tags," in *Proc. 3rd Annu. Int. Conf. Mobile Ubiquitous Syst.*, Jul. 2006, pp. 1–4.

[32] P. Peris-Lopez, A. Orfila, A. Mitrokotsa, and J. C. A. van der Lubbe, "A comprehensive RFID solution to enhance inpatient medication safety," *Int. J. Med. Informat.*, vol. 80, no. 1, pp. 13–24, Jan. 2011.

[33] M. H. Ozcanhan, G. Dalkilic, and S. Utku, "Analysis of two protocols using EPC Gen-2 tags for safe inpatient medication," in *Proc. IEEE INISTA*, Jun. 2013, pp. 1–6.

[34] Y. Lien, X. Leng, K. Mayes, and J.-H. Chiu, "Reading order independent grouping proof for RFID tags," in *Proc. IEEE Int. Conf. Intell. Secur. Informat.*, Jun. 2008, pp. 128–136.

[35] S. Jantarapatin, C. Mitrpant, C. Tantibundhit, T. Nuamcherm, and P. Kovintavewat, "Performance comparison of the authentication protocols in RFID system," in *Proc. Int. Conf. Manage. Emergent Digit. EcoSystems*, 2010, pp. 131–136.

[36] T. Nuamcherm, P. Kovintavewat, C. Tantibundhit, U. Ketprom, and C. Mitrpant, "An improved proof for RFID tags," in *Proc. 5th Int. Conf. Electr. Eng., Electron., Comput., Telecommun. Inf. Technol.*, May 2008, pp. 737–740.

[37] Y.-C. Yen, N.-W. Lo, and T.-C. Wu, "Two RFID-based solutions for secure inpatient medication administration," *J. Med. Syst.*, vol. 36, no. 5, pp. 2769–2778, Oct. 2012.

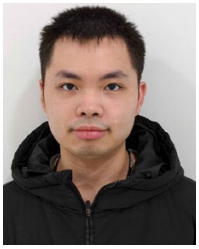
[38] J. Hermans and R. Peeters, "Private yoking proofs: Attacks, models and new provable constructions," in *Proc. Int. Workshop Radio Freq. Identificat., Secur. Privacy Issues*. Cham, Switzerland: Springer, 2012, pp. 96–108.

[39] H.-M. Sun, W.-C. Ting, and S.-Y. Chang, "Offlined simultaneous grouping proof for RFID tags," in *Proc. 2nd Int. Conf. Comput. Sci. Appl.*, Dec. 2009, pp. 1–6.

[40] G.-F. Shen, S.-M. Gu, and D. Liu, "An anti-counterfeit complete RFID tag grouping proof generation protocol," *Int. J. Netw. Secur.*, vol. 21, no. 6, pp. 889–896, 2019.
[41] R. G. Protocol, V. Cherneva, and J. L. Trahan, "A secure and efficient parallel-dependency RFID grouping-proof protocol," *IEEE J. Radio Freq. Identificat.*, vol. 4, no. 1, pp. 14–23, Mar. 2020.
[42] J. M. de Fuentes, P. Peris-Lopez, J. E. Tapiador, and S. Pastrana, "Probabilistic yoking proofs for large scale IoT systems," *Ad Hoc Netw.*, vol. 32, pp. 43–52, Sep. 2015.
[43] I. O. F. Standardization. *ISO/IEC 18000: Information Technology Automatic Identification and Data Capture Techniques-Radio Frequency Identification for Item Management Air Interface*. Accessed: Oct. 22, 2021. [Online]. Available: https://www.iso.org/
[44] K.-Y. Tsai, M. Yang, J. Luo, and W.-T. Liew, "Novel designated ownership transfer with grouping proof," *Appl. Sci.*, vol. 9, no. 4, p. 724, Feb. 2019.
[45] Z. Sun and M. Wei, "PUF-based anonymous RFID system ownership transfer protocol," in *Proc. Chin. Control Conf. (CCC)*, Jul. 2019, pp. 6367–6373.
[46] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar, "A survey on privacy protection in blockchain system," *J. Netw. Comput. Appl.*, vol. 126, pp. 45–58, Jan. 2019.
[47] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 839–858.
[48] *EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID; Specification for RFID Air Interface Protocol for Communications at 860 MHz–960 MHz*, EPCglobal, Nov. 2013, vol. 1, no. 3, p. 14.
[49] M. Sidorov, M. T. Ong, R. V. Sridharan, J. Nakamura, R. Ohmura, and J. H. Khor, "Ultralightweight mutual authentication RFID protocol for blockchain enabled supply chains," *IEEE Access*, vol. 7, pp. 7273–7285, 2019.
[50] *Homepage | GS1*. Accessed: Oct. 1, 2021. [Online]. Available: https://www.gs1.org/
[51] H.-Y. Chien and S.-B. Liu, "Tree-based RFID yoking proof," in *Proc. Int. Conf. Netw. Secur., Wireless Commun. Trusted Comput.*, vol. 1, Apr. 2009, pp. 550–553.
[52] M. J. Dworkin et al., "Advanced encryption standard (AES)," NIST, FIPS, Gaithersburg, MD, USA, 2001.
[53] Public Key Cryptography Standard, *PKCS #1 v2.2: RSA Cryptography Standard*, RSA Laboratories, Oct. 2012.
[54] *Draft Amendment to STANDARD FOR Telecommunications and Information Exchange Between Systems-LAN/MAN Specific Requirements—Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Security Enhancements*, IEEE Standard 802.11 i/D7, 2003.
[55] L. Gong, R. Needham, and R. Yahalom, "Reasoning about belief in cryptographic protocols," in *Proc. IEEE Comput. Soc. Symp. Res. Secur. Privacy*, Jun. 1990, pp. 234–248.
[56] A. Poschmann, G. Leander, K. Schramm, and C. Paar, "New light-weight crypto algorithms for RFID," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 1843–1846.
[57] A. S. W. Man, E. S. Zhang, V. K. N. Lau, C. Y. Tsui, and H. C. Luong, "Low power VLSI design for a RFID passive tag baseband system enhanced with an AES cryptography engine," in *Proc. 1st Annu. RFID Eurasia*, Sep. 2007, pp. 1–6.

MING-HOUR YANG (Member, IEEE) received the Ph.D. degree in computer science and information engineering from the National Central University, Taiwan. His research interests include network security and system security with particular interests on security issues in RFID and NFC security. Topics include: mutual authentication protocols, secure ownership transfer protocols, polymorphic worms, and tracing mobile attackers.

YU-SHAN HSU received the master's degree in information and computer engineering from Chung Yuan Christian University, where she is currently pursuing the Ph.D. degree with the Department of Electrical Engineering.



HUNG-YU KO received the master's degree in information and computer engineering from Chuag Yuan Christian University.