Sensors **2015**, *15*, 27087-27115; doi:10.3390/s151027087

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

A Novel Multilayered RFID Tagged Cargo Integrity Assurance Scheme

Ming Hour Yang [1,*], Jia Ning Luo [2] and Shao Yong Lu [1]

[1] Chung Yuan Christian University, Chung Pei Road, Chung Li District, Taoyuan 32023, Taiwan; E-Mail: lumosac@gmail.com

[2] Ming Chuan University, Deming Road, Guishan District, Taoyuan 33348, Taiwan, E-Mail: deer@mail.mcu.edu.tw

* Author to whom correspondence should be addressed; E-Mail: mhyang@cycu.edu.tw; Tel.: +886-3-2654-733; Fax: +886-3-2654-799.

Academic Editor: Vittorio M. N. Passaro

Received: 11 August 2015 / Accepted: 19 October 2015 / Published: 23 October 2015

Abstract: To minimize cargo theft during transport, mobile radio frequency identification (RFID) grouping proof methods are generally employed to ensure the integrity of entire cargo loads. However, conventional grouping proofs cannot simultaneously generate grouping proofs for a specific group of RFID tags. The most serious problem of these methods is that nonexistent tags are included in the grouping proofs because of the considerable amount of time it takes to scan a high number of tags. Thus, applying grouping proof methods in the current logistics industry is difficult. To solve this problem, this paper proposes a method for generating multilayered offline grouping proofs. The proposed method provides tag anonymity; moreover, resolving disputes between recipients and transporters over the integrity of cargo deliveries can be expedited by generating grouping proofs and automatically authenticating the consistency between the receipt proof and pick proof. The proposed method can also protect against replay attacks, multi-session attacks, and concurrency attacks. Finally, experimental results verify that, compared with other methods for generating grouping proofs, the proposed method can efficiently generate offline grouping proofs involving several parties in a supply chain using mobile RFID.

Keywords: radio-frequency identification; multilayered grouping proof; supply chain management; anonymity; multi-session attacks; concurrency attacks

1. Introduction

Mobile radio frequency identification (RFID), a widely adopted technique in supply chain management (SCM), involves using a mobile reader to scan RFID tags [1,2]. In each SCM stage, RFID tags can be employed to store cargo information and facilitate autonomous stocktaking, thereby improving inventory management and accelerating the retail cycle [3]. RFID tags can also be employed to track and locate cargo, which not only improves order management [3], but also reduces management costs. For example, by adopting an RIFD SCM system, Wal-Mart reduced its annual distribution costs by approximately 6%–7%, an equivalent saving of US$1.4 billion [4].

Although tracking cargo through RFID is easier than using other logistics management methods, malicious members of a supply chain can steal cargo during the transfer process [5,6]; consequently, delivery of all cargo cannot be guaranteed. Annually, cargo theft accounts for approximately US$30 billion in losses worldwide [4]. However, when determining whether the suppliers, transporters, and recipients are suspects, disputes may arise when the actual suspect cannot be confirmed [7]. To solve this problem, Zhou and Rodrigues have proposed a smart grid based infrastructure to verify the generated codes for each recipient [8]. Lo *et al.* [9] proposed a cargo tracking system utilizing RFID. The tracking system provides the ability to report the real-time locations of the cargo and its flow. RFID systems have been designed to generate grouping proofs for all RFID tags during cargo transfer [10–12]. Accordingly, in the event of a dispute, transporters can use the receipt proofs as evidence of having delivered the cargo. Moreover, by generating a receipt proof when the cargo is delivered, the recipient provides undeniable proof that they have received their shipment in full.

However, if a poorly designed integrity check mechanism is used, malicious users can exploit the mechanism, thereby compromising its credibility. For example, attackers can record the messages communicated between tags and readers and then replay them to generate bogus proofs that can pass authentication protocols, despite nonexistent tags being used in the attack. Saito and Sakurai [13] employed timestamps generated by a verifier to replace random numbers [14] to protect against replay attacks. However, tag impersonation attacks can be used to exploit generating timestamps in systems that generate multiple grouping proofs simultaneously. Therefore, Peris-Lopez *et al.* [15,16] have proposed a protocol for generating grouping proofs that protect against tag impersonation attacks and safeguards the method proposed by Yu, Hou, and Chiang [17] by not only protecting against multisession attacks, in which malicious users intercept pieces of grouping proofs transmitted in the same time zone and then generate a composite bogus proof, but also protecting RFID systems from most attacks targeting online grouping proofs. However, because generating the grouping proof protocol requires writing data to an RFID tag multiple times, concurrency attacks can occur when multiple readers concurrently generate grouping proofs for the same RFID tag, causing messages from the readers to interactively overwrite the tag content generated by each reader in the tag, thus generating the erroneous grouping proofs. Therefore, Lin, Lai, Tygar, Yang, and Chiang [18] proposed a protocol that instantly generates grouping proofs to protect against concurrency attacks that prevent grouping proofs from being generated.

When transporters receive or deliver cargo in areas where mobile readers cannot connect to the verification server (verifier) for authentication, the server cannot generate a trustworthy time value to initiate generating the grouping proofs [10]. Therefore, Sun, Ting, and Chang [19] and Hermans and Peeters [20] have utilized the timeout mechanism of RFID tags to ensure that the grouping proofs are

accurate. When group members cannot complete the protocol because they are not within range of a mobile reader, the reader cannot obtain the grouping proofs or ensure that all of the RFID tags are in the same interval when the proofs are generated. Lo and Yeh [21] and Ma, Lin, Wang, and Shang [22] have employed additional active tags or trusted devices that generate timestamps to ensure that the time difference between timestamps is within a time threshold before responding to a mobile reader's requests for the generated proofs; thus, all the of the RFID tags involved in the grouping proofs must be present at the same interval.

In addition to the correctness of the generated grouping proofs, the conventional grouping proofs [14,23–28] are generated by all tags on site one after another, thus making these tags incapable of simultaneously computing the pieces of proof assigned to the tags. The calculation time for generating this type of grouping proof increases with the number of tags; thus, it is unsuitable for cargo with a considerable number of RFID tags, which is common in most supply chains. Lien, Leng, Mayes, and Chiu [29] proposed a method that satisfies the exclusiveness of commutative property or is capable of combining the pieces of proof computed by the tags through XOR operands in order to generate grouping proofs. The method enabled the reader to request that proofs be generated by other tags before the pieces of proof have been reconstructed. Without needing to consider the sequence of requests, the request order can be directly combined to generate the final grouping proof. In addition, parallel computing accelerates the generation of grouping proofs. Jantarapatin, Mitrpant, Tantibundhit, Nuamcherm, and Kovintavewat [30] and Nuamcherm, Kovintavewat, Tantibundhit, Ketprom, and Mitrpant [31] have improved the security of message requests for grouping proofs by enabling mobile readers to directly broadcast the random number incorporated in the tags. All tags must simultaneously use both the key shared by the verification server and encrypt to obtain the random number sent by the reader in order to obtain protocol for generating the pieces of proof. Similarly, the final grouping proof is produced by reconstructing the pieces of proof generated by each tag through a process of exclusion or combination, which reduces the amount of time taken for generating the proof. However, with this method, compromised readers can be exploited to insert random numbers in the request messages before transmitting them to the tags; after recovering the pieces of proof from the tags, a piece of proof with a tag that is the same as the tag of the inserted random number can be used to generate bogus grouping proofs that can pass authentication, despite not having the tags. Thus, Sun, Ting, and Chang [19] proposed using message broadcast to render it impossible for readers to generate a bogus grouping proof with nonexistent tags through encrypting the returned messages of all the participating tags using a shared key. Yen, Lo, and Wu [32] indicated that when a reader is online (*i.e.*, connected to a network), all tags should first be authenticated, and the grouping proofs should then be generated and transmitted to a backend verifier; thus, completion can be achieved by broadcasting only two messages. Hermans and Peeters [20] employed an additional timestamp to initiate the procedure of generating grouping proofs and signing the final collected grouping proof, thus achieving the same outcome except in an offline environment.

The time taken for generating pieces of proof can be reduced considerably by using parallel computing because the requests for grouping proofs are broadcast to all RFID tags. However, when these tags transmit the pieces of proof to the reader, collisions occur because of simultaneously receiving a high number of tag messages. If messages are sent using anticollision mechanisms, such as tree-walking or the aloha protocol of ISO-18000 [33], then the time taken for responding to messages

increases [34,35]; thus, malicious users can exploit this time difference to generate a legitimate grouping proof from a group of tags that are not in the same time interval [36]. Therefore, Leng, Lien, Mayes, and Markantonakis [37] proposed a subgrouping grouping proof that minimized the number of collisions by using fewer grouping tags. This method involves dividing groups of tags into several subgroups and generating proofs sequentially; however, because only one reader is employed to generate the grouping proofs, retransmitting messages across subgroups can only be performed in the specific sequence; thus employing parallel retransmission does not improve the effectiveness. Moreover, the scanning restrictions on the maximum number of tags that can be read by the reader [38,39] render this method unfeasible in logistics management when a high number of tags requiring scanning are involved.

Therefore, the present study proposed a method for generating multilayered grouping proofs to resolve the problem of needing to scan generated grouping proofs in batches because of the limited number of tags that can be scanned using a reader [38,39], which, in a supply chain environment, would invalidate the grouping proof because the processing time would exceed its time threshold. The goal of our method is proposing a hierarchical management framework that enables several readers to simultaneously generate pieces of proof; subsequently, the final grouping proofs are generated by an authorized reader**.** In addition, regardless of whether the reader can connect to the backend verifier, the cargo transporter can use the reader to collect tag information to generate proofs. When the messages are transmitted in an online environment, the verifier can confirm whether the tags were generated by readers in the same location. Thus, the proposed method is effective for (1) generating grouping proofs for a supply chain in which a considerable number of tags must be scanned simultaneously; (2) reducing the time required for generating proof concurrently through multiple readers when a high number of tags are involved; (3) providing undeniable proof of delivery for recipients, suppliers, and transporters by generating proofs before every cargo transfer and by acquiring signatures from all relevant personnel as evidence that the cargo was received in full and by the intended recipient; (4) ensuring that the tags are secure from replay, eavesdropping, and impersonation attacks, and as well as most attacks designed to generate bogus grouping proofs; (5) ensuring data confidentiality through using anonymous tags; (6) ensuring that the supply chain's cargo routes remain confidential; (7) automatically determining whether the transporters deliver the cargo on time and according to order; and (8) dynamically scaling the number of cargo tags according to the scan rate at which the mobile reader scans to generate grouping proofs.

The remainder of this paper is organized as follows. Section 2 introduces the environmental hypothesis for the grouping-proof protocol applied to many RFID readers, as well as the relationships among the tags, readers, and certain members in a supply chain. In addition, the initialization process and methods for generating and authenticating grouping proofs are explained in detail. Subsequently, Section 3 investigates the security of the proposed protocol and compares it with that of other grouping proofs. In Section 4, the computation effectiveness of the proposed method is analyzed and compared with that reported by related studies. Finally, Section 5 offers the conclusions for this study.

2. Cargo Inspection Management of Mobile Logistics

The method proposed in this study can be applied to improve SCM. As depicted in Figure 1a, the proposed method can be adopted to automatically generate grouping proofs to facilitate using mobile

RFID readers to manage cargo. To automate the scanning process for transporters delivering cargo, suppliers, transporters, and recipients adopting the protocol must first register their mobile RFID devices on a verifier (*i.e.*, a server for backend authentication). In addition, the threat model developed is based on the hypothesis that malicious users can conduct eavesdropping attacks by intercepting the messages transmitted by RFID readers and tags. In addition, transporters might exploit the proposed protocol through the following two malicious behaviors: (1) providing a bogus proof to conceal stolen cargo; and (2) tampering with the proof timestamp to conceal delayed deliveries caused by negligence. Therefore, in this study, a clock tag was incorporated into the transporter's reader to ensure that the timestamp generated under offline conditions is trustworthy for when the reader cannot connect to the verifier.

As indicated by Step 1 in Figure 1a, when the transporter requests the verifier to authenticate the delivery of cargo to *n* recipients, denoted as $P^1, P^2, \ldots, P^n$, the delivery of cargo to any recipient can be expressed as shown in Equation (1):

$$T^q = \left\{ TID_i^q \mid \forall i \; TID_i^q \in P^q, 1 \le i \le \delta q, \delta q \in \mathbb{Z}^+ \right\} \quad (1)$$

where TID_1^q,TID_2^q,…,$TID_{\delta q}^q$ denote the RFID tag codes; δq represents the number of RFID tags; and PID^q is the recipient's RFID code, which is incorporated into the tags.

Assume that a supplier must deliver cargo with a group of RFID tags, denoted as $T^0 = T^1 \cup T^2 \cup \ldots \cup T^n$, where *n* denotes the number of recipients. To ensure that the transporter can immediately check the integrity of cargo delivered to recipient P^q, the verifier applies Equation (2) to obtain a verification code indicating the integrity of the delivered cargo:

$$TA^q = TH_1^q || TH_2^q || \ldots || TH_{\delta q}^q, where \; \forall i \; TH_i^q = H\left(TID_i^q || Kt_i^q || TS_v\right) \quad (2)$$

where TA^q is the verification code; Kt_i^q denotes the shared key for the verifier and the tag TID_i^q; and TS_v is a timestamp generated by the verifier.

To enable the RFID reader to encrypt the multicast messages transmitted to the cargo tags and to establish a secure multicast channel between the readers and tags [40], the verifier generates the group keys for the *k*-ary tree with GK_0^q as the starting node (a detailed explanation of key tree is provided in Section 2.1). At Step 2 in Figure 1a, the verifier transmits the following data to the transporter's reader: a group of verification codes $TA^0 = \{TA^1, TA^2, \ldots, TA^n\}$ for *n* recipients; a key tree comprising the group of tags $GK_0^0 = \{GK_0^1, GK_0^2, \ldots, GK_0^n\}$ from the recipients; the timestamp TS_v; and verification codes for the clock tags generated in the offline phase. Figure 1b shows that the reader generates verification codes for the transporter, supplier, all of the RFID tags involved in delivering the cargo, and a receipt proof for the cargo delivery. Subsequently, the check codes generated by the transporter's reader and the verifier are compared to confirm whether they are identical in order to ensure the integrity of the cargo and that it has been received in full.

As indicated in Figure 1c, when the transporter delivers cargo to each recipient, the cargo tags and recipient's RFID tag are scanned using the reader, which then generates receipt proofs and verification codes for each recipient. The integrity and correctness of the delivered cargo are confirmed by ensuring that the verification code issued by the verifier and the receipt proof and verification code generated by the transporter are consistent. The reader transmits the pick proofs and receipt proofs to the verifier as soon as a connection is established. In the event of a dispute (e.g., the recipient denying having received the cargo), the receipt proof is evidence that the transporter has already delivered the cargo in full, as

illustrated in Step 6 of Figure 1a. By contrast, if the transporter denies that the cargo has been consigned by the supplier, then the pick proof is evidence that the transporter has already collected the cargo, as indicated in Step 4 of Figure 1a. Moreover, if the recipient notices that the cargo content differs from the consignment note, then the backend verifier can cross-reference the pick proof and receipt proof. Should the tags for the receipt proofs and pick proofs be identical, then the error is associated with the quantity shipped by the supplier. A discrepancy between the receipt proof and pick proof indicates that the transporters has lost cargo in route. Thus, the grouping proofs solve and clarify problems regarding lost cargo.

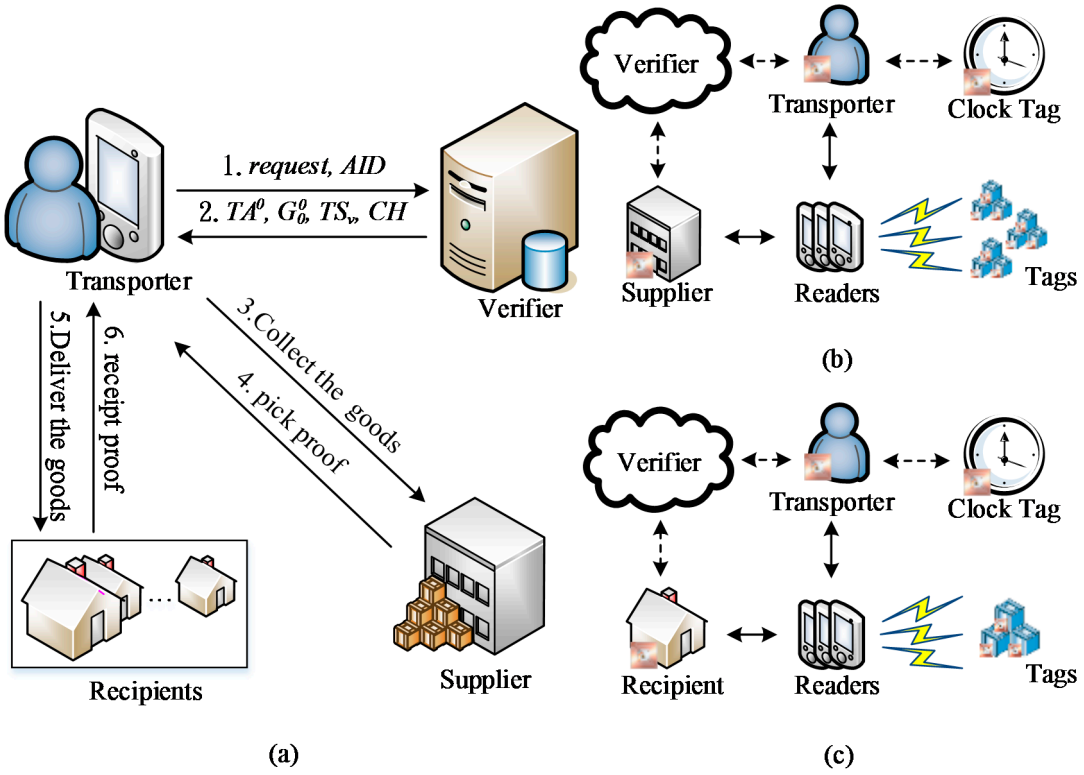


Figure 1. Process of supply chain distribution and grouping proof: (**a**) steps of the supply chain distribution; (**b**) generate the grouping proofs for pick up and delivery; (**c**) generate the grouping proofs for cargo delivery.

However, some problems remain unresolved in this method of logistics verification; when the reader cannot connect to the verifier, trusted proofs cannot be generated securely. Regarding the method proposed in this study, identical procedures are used for picking up cargo from suppliers and transporting cargo to recipients. Although pick proofs must be generated from cargo tags when suppliers consign cargo for deliver to multiple recipients, the method proposed in this study is no different except for the number of tags, the personnel engaged in the process, and the identification code of the reader. Without loss of generality, the subsequent sections discuss the following three phases for generating grouping proofs, which is achieved through the transporter and recipient using the identification codes AID and PID^q, respectively: (1) an initialization phase for generating the group key tree; (2) an integrity verification phase for generating grouping proofs and integrity check codes for when the multilayered reader is used in offline phase; and (3) a dispute resolution phase, in which all of the grouping proofs are examined to verify the cargo delivery process in the event of a dispute.

2.1. Initialization Phase

When a cargo shipment with a tag collection T^q is delivered to recipient P^q, the reader requests the verifier to establish a secure multicast connection to ensure that the generated grouping proofs accord with those on the recipient's reader and that messages can be transmitted to the δq tags in T^q. Therefore, the verifier generates a k-ary group key tree with a height difference of the subtree of ≤1 (tree height of $h_{max} = \lceil \log_k(\delta q/k) \rceil$) for the shared key Kt_i^q available to all tags TID_i^q. In summary, the group keys that can transmit messages to δq tags in T^q are defined as GK_0^q, as illustrated in Figure 2. Figure 2a shows the numbering sequence that is generated when the group number of a certain node in the k-ary group key tree is G_s^q: from top to bottom, from left to right, the number of parent nodes is $G^q_{\lfloor (s-1)/k \rfloor}$, and the number of the child nodes ranges from G^q_{s*k+1} to G^q_{s*k+k}. Figure 2b indicates an example of a 3-ary key tree generated by T^q, a set of 23 tags; the group key GK_2^q is employed to encrypt the multicast messages transmitted to the tags numbering from TID_{10}^q to TID_{18}^q, and the tags TID_{10}^q, TID_{11}^q, and TID_{12}^q decrypt the multicast messages encrypted with the group key GK_7^q by using the keys Kt_{10}^q, Kt_{11}^q, and Kt_{12}^q shared with the verifier.

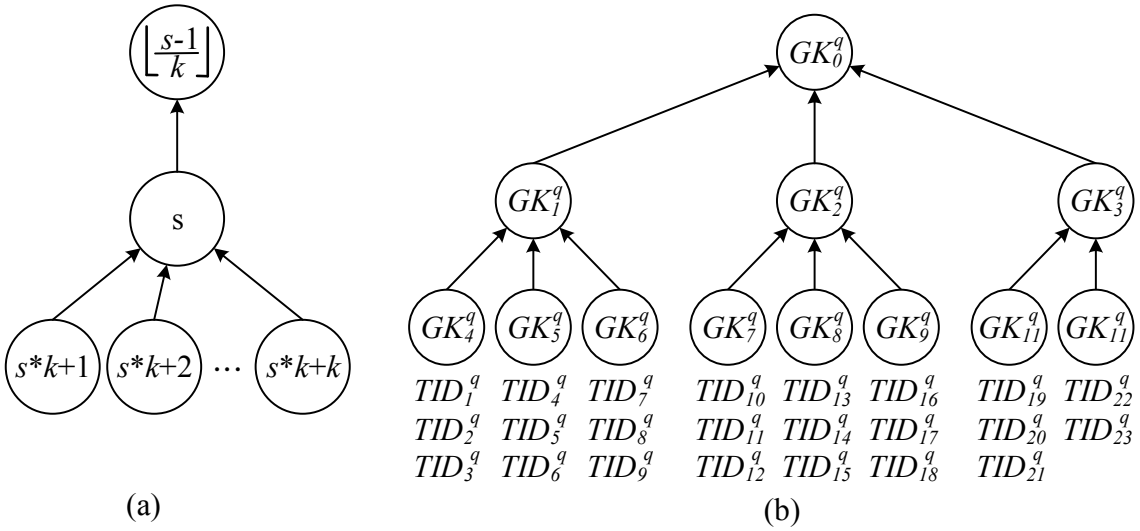


Figure 2. Group Key tree of cargo tags of recipient P^q: (**a**) Rule for numbering among group keys; (**b**) key tree for 3-ary group tags.

Therefore, the nodes with the group number of G_s^q are composed by 1 to a number of k subtrees, with Equation (3) indicating the group key incorporated by any node G_s^q. Group key GK_s^q exists in all the parent groups numbered G_{spar}^q that incorporate G_s^q and satisfy the intersection of group number G_s^q and G_{spar}^q equaling G_s^q. The difference set between G_0^q and G_{spar}^q and the intersection between G_0^q and G_s^q as an empty set is as indicted in Equation (4).

$$G_s^q = \left\{ GK_l^q \,\middle|\, \forall l\, GK_l^q \in G_s^q, s * k^h + \frac{k^h - 1}{k-1} \le l \le s * k^h + \frac{k(k^h-1)}{k-1}, h \in \mathbb{Z}_0{}^+, s \in \mathbb{Z}_0{}^+ \right\}$$

$$G_{spar}^q = \left\{ GK_s^q \in G^q_{\left\lfloor \frac{s - \frac{k^h-1}{k-1}}{k^h} \right\rfloor}, h \in \mathbb{Z}_0{}^+, s \in \mathbb{Z}_0{}^+ \right\}, \quad \forall s\, GK_s^q, \text{where } G_s^q \cap G_{spar}^q = G_s^q \tag{3}$$

and

$$\left(G_0^q - G_{spar}^q\right) \cap G_s^q = \emptyset \tag{4}$$

Table 1. Definition of symbols.

V	$Verifier$: a third-party verification server to reinspect grouping proofs
S	$Supplier$: shipping supplier
A	$Transporter$: transporter who delivers cargo
P^q	$Recipient$: q-th recipient who receives the cargo
C	$Clock\ Tag$: a third-party clock tag providing time for the system in offline phase
AID	Identification code for A
CID	Identification code of a trusted and active third-party C
RID_0	Identification code of the reader used by a
PID^q	Identification code of P^q
RID_i^q	Identification code of the j-th reader used by P^q
TID_i^q	Identification code of the i-th tag for P^q
G_s^q	s-th group code for P^q
TH_i^q	Verification hash value for TID_i^q
Kc	Shared key for C and V
Kr_i^q	Shared key for RID_i^q and V
Kt_i^q	Shared key for TID_i^q and V
GK_s^q	Shared key for G_s^q and V
SK_i^q	Session key among readers
PK^a	Public key for a
PR^a	Private key for a
PK^q	Public key for P^q
PR^q	Private key for P^q
Nr_i^q	Random number generated by RID_i^q
Nt_i^q	Random number generated by TID_i^q
Nc	Random number generated by C
Na	Random number generated by a
Np^q	Random number generated by P^q
TS_v	Timestamp generated by V
TS_c	System time of C
ΔT	Time threshold for generating grouping proofs
$E(key, Msg)$	Encryption function generated by message (Msg) through an employment of symmetric key (key)
$Sign(key, Msg)$	Signing function generated by Msg through an employment of private key (key)
$MAC(key, Msg)$	Function for message authentication code generated by Msg by an employment of Key
$H(Msg)$	Message authentication code generated from an employment of hash function by Msg
FO^q	Judgement of whether the grouping proof for P^q is generated in online or offline phase

In addition, this study defined the $\lceil \delta q/k \rceil$ -number of key trees coded from $\left\lceil \frac{(\delta q/k)-1}{k-1} \right\rceil$ to $\left\lceil \frac{(\delta q/k)-1}{k-1} \right\rceil + \lceil \delta q/k \rceil - 1$, that directly connects with the nodes on the tags, as leaf group, as indicated in Equation (5) and represented by $G^q_{leaf,m}$. For example, tag codes TID^q_1, TID^q_2, and TID^q_3 connect with the leaf node $G^q_{leaf,1}$ of group code G^q_4.

$$G^q_{leaf,m} = \left\{ TID^q_l \,\middle|\, \forall l\, TID^q_l \in G^q_{leaf,m}, (m-1)k+1 \le l \le mk, 1 \le m \le \left\lceil \frac{\delta q}{k} \right\rceil \right\} \quad (5)$$

Table 1 provides the definition of the symbols used in the protocol.

2.2. Integrity Verification Phase: Grouping Proof Protocol of a Multilayered Reader

After the transporter delivers the cargo to the recipient and simultaneously generates grouping proofs using the reader with a maximum reading capacity of *r*, the group keys are distributed to several mobile readers from the transporters' reader RID_0 to securely multicast messages to δq tags via the recipients' readers, thus enabling each reader to receive the maximum number of tags by performing only one multicast; in other words, the grouping proof is generated using the minimum number of group keys. During the initialization phase, Equation (3) is employed to generate the group keys and construct a complete *k*-ary key tree for RID_0 with a tree height of $h = \lceil \log_k(\delta q/k) \rceil - \lfloor \log_k(r/k) \rfloor$ for multicasting $k^{\lfloor \log_k r \rfloor}$ tags; in other words, the key tree satisfies the maximum number of reading limits, *r*, and can encrypt the maximum number of tags with group keys, thereby forming a complete subtree. Therefore, when the number of tags that k^h group key (whose height h equals that of the key tree) can encrypt ($k^h \times k^{\lfloor \log_k r \rfloor}$ tags) equals δq, the complete key tree with successive group keys with a multicast range from 1 to r for multicasting $k^{\lfloor \log_k r \rfloor}$ tags is derived, which also satisfies the nonrepetition requirement for the tags.

$$f(a) = \left\lfloor \frac{\left\lceil \frac{(\delta q/k)-1}{k-1} \right\rceil + \lceil \delta q/k \rceil - 1 - \frac{k^a - 1}{k-1}}{k^a} \right\rfloor \quad (6)$$

However, if $k^h \times k^{\lfloor \log_k r \rfloor} > \delta q$, then the *k*-ary key tree (Figure 3) would be incomplete. To enable the reader to scan all of the tags with the least number of group keys, the system must perform a search to determine whether any incomplete tree contains a group key that can encrypt a particular number of tags, the number ranging between $k^{\lfloor \log_k r \rfloor}$ and *r*. Because an incomplete subtree would appear in the subtree with the maximum number of leaf nodes, Equation (6) is applied to determine $f(\lfloor \log_k r \rfloor)$ in order to determine the parent node code at the level of $\lfloor \log_k r \rfloor$ (height = $h-1$) above the maximum leaf node code level $\left\lceil \frac{(\delta q/k)-1}{k-1} \right\rceil + \lceil \delta q/k \rceil - 1$. Subsequently, Equation (3) is employed to determine all of the leaf group codes in the encrypted nodes; also in addition, Equation (5) is applied to compute *RA*, the total number of tags that can be read. As expressed in Equation (7), when the maximum number of *r* tags scanned by the reader can contain *RA* number of tags, a group key with the code $f(\lfloor \log_k r \rfloor)$, which satisfies the number of tags between $k^{\lfloor \log_k r \rfloor}$ and *r* quantity, is obtained. Consequently, the group key with the code $f(\lfloor \log_k r \rfloor)$ is selected as the first key R_f; else, the smallest code in which the message can multicast to $k^{\lfloor \log_k r \rfloor}$ number of tags can be selected as the first key R_f.

Subsequently, Equation (8) was employed to compute that under these two conditions, another group key code R_e larger than R_f, enabling the key with codes ranging between R_f and R_e to be capable of encrypting all tags, with none of the two keys repeatedly encrypting the same tag.

$$R_f = \begin{cases} f(\lfloor \log_k r \rfloor), RA \le r \\ f(\lfloor \log_k r \rfloor) + 1, RA > r \end{cases} \quad (7)$$

$$R_e = \begin{cases} k * R_f, k * R_f < \left\lceil \frac{\lfloor \delta q/k \rfloor - k}{k-1} \right\rceil + \lceil \delta q/k \rceil - 1 \\ \left\lceil \frac{\lfloor \delta q/k \rfloor - 1}{k-1} \right\rceil + \lceil \delta q/k \rceil - 1, k * R_f \ge \left\lceil \frac{\lfloor \delta q/k \rfloor - k}{k-1} \right\rceil + \lceil \delta q/k \rceil - 1 \end{cases} \quad (8)$$

For example, Figure 3a is a distribution diagram of Figure 2b that involves the results obtained from the 23 group keys tagged on the k = 3-ary group key tree for the reader with a reading capacity of r = 6. According to $\lceil \log_3(23/3) \rceil - \lfloor \log_3(6/3) \rfloor$=2-0=2, the group key with the tree height of 2 can multicast message to three tags and encrypt the maximum number of tags within the capacity that can be read by the reader. In addition, of the parent node located in the level of $\lfloor \log_3 6 \rfloor = 1$ above the group key GK_{11}^q of the largest leaf group within an incomplete key tree coded 11, the leaf group keys GK_{10}^q and GK_{11}^q incorporated in key GK_3^q, with a tree height of 1, can read three and two tags, respectively. This study computed that an incomplete subtree with the height of 1 can read a total of 5 tags, which was less than 6, the number of tags that can be simultaneously read by a reader; in addition, the tags that can be encrypted by the largest leaf node was already incorporated within the group key GK_3^q ($k \times R_f = 3 \times f(2)$ =9→9 < 11). Therefore, a reader generates grouping proofs from the group keys coded from 3 to 9 (GK_3^q to GK_9^q).

Figure 3b is a similar distribution diagram of Figure 2b as that of Figure 3a as it presents the results from the group key to the reader on the key tree; however, Figure 3b differs from Figure 3a in that the reader had a reading capacity of r = 4. In this example, the starting group key GK_3^q of an incomplete subtree with the height of 1 can read five tags, which was larger than the number of tags that can be simultaneously read by a reader. Therefore, the group key GK_3^q was replaced by the subgroup keys GK_{10}^q and GK_{11}^q and the reader with the reading capacity of four tags generated grouping tags using group keys ranging from GK_4^q to GK_{11}^q.

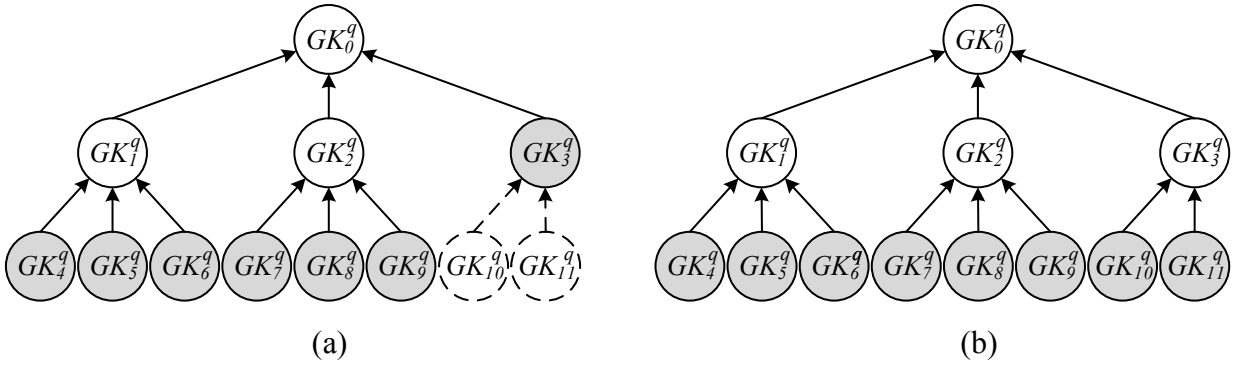


Figure 3. Examples showing group key selection in an incomplete group Tree: (**a**) the number of remaining tags $RA \le$ reading capacity of r tags; (**b**) $RA > r$.

Subsequently, the following two strategies were employed to distribute $R_e - R_f + 1$ number of group keys to the reader with a reading capacity of r.

I. Only one group key was distributed to each reader used by a recipient to maximize the benefit of concurrent reading; consequently, a total of $\left\lceil \frac{R_e - R_f}{r-1} \right\rceil + R_e - R_f$ readers was required to generate the grouping proofs.

II. Distribute one or multiple group keys to the reader to satisfy the condition that within all the readers, the total number of tags that can be encrypted by the key was less than the number of *r* tags, and only a minimal number of readers for recipients was required [41–43]; therefore, grouping proofs were generated using the least resources.

Finally, when the number of recipients' readers was larger than *r*, reader RID_0 cannot simultaneously transmit messages to all recipients' readers. The method as indicated in Figure 2a in Section 2.1 was thus employed to form a read tree and code reader to solve the problem in which several readers simultaneously generate grouping proofs. To ensure the security of group key transmission and message transmission among readers, any reader RID_j^q can use the key shared with the verifier to generate a session key SK_j^q for encrypting messages transmitted between the parent node as well as to generate a maximum number of *r* keys (ranging from SK_{j*r+1}^q to SK_{j*r+r}^q) for child node communication, and for encrypting messages transmitted between two readers. For example, Figure 4 presents the read tree in which each or several of the seven group keys GK_3^q, GK_4^q, GK_5^q, GK_6^q, GK_7^q, GK_8^q, and GK_9^q, as indicated in Figure 3a, were distributed to each reader. Figure 4a indicates that when the seven keys were processed into seven readers to enable all readers to concurrently read all the tags, transporter's reader RID_0 with a reading capacity for only six tags could not simultaneously transmit messages to seven readers owned by the recipients; instead, a middle reader was required for transferring messages. Thus, eight ($\left\lceil \frac{R_e - R_f}{6-1} \right\rceil + R_e - R_f = 8$) readers were required in total. Pieces of proof were generated by RID_2^q to RID_8^q; among them, reader RID_7^q generated pieces of proof with tag codes TID_{13}^q, TID_{14}^q, and TID_{15}^q from the distributed group key GK_8^q, with the proofs encrypted by session key SK_7^q and transmitted back to the reader for decryption by parent node RID_1^q. Moreover, Figure 4b presents the results when several keys were written into the same reader; thus, only four readers (RID_1^q to RID_4^q) were needed to simultaneously generate the pieces of proof for all tags. However, because a single reader such as RID_4^q distributed two group keys GK_8^q and GK_9^q, thus six pieces of proof for tags coded from TID_{13}^q to TID_{18}^q must be generated.

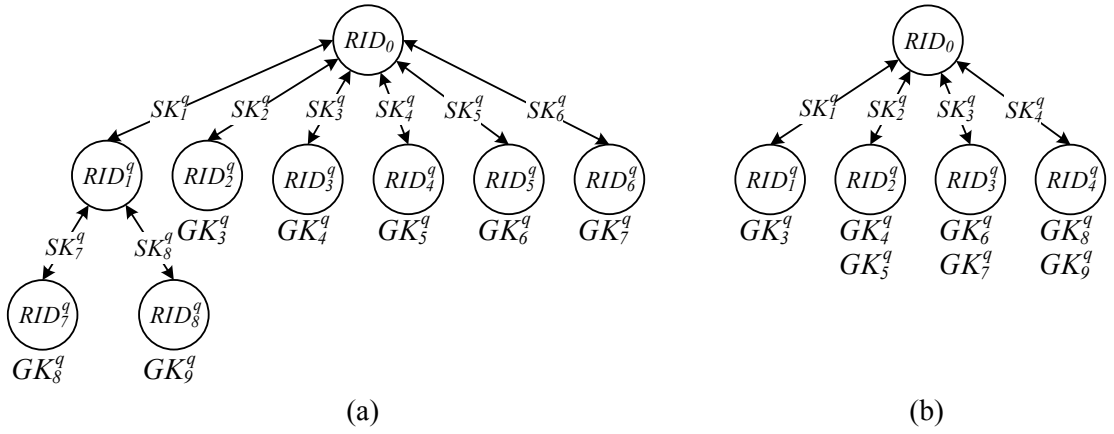
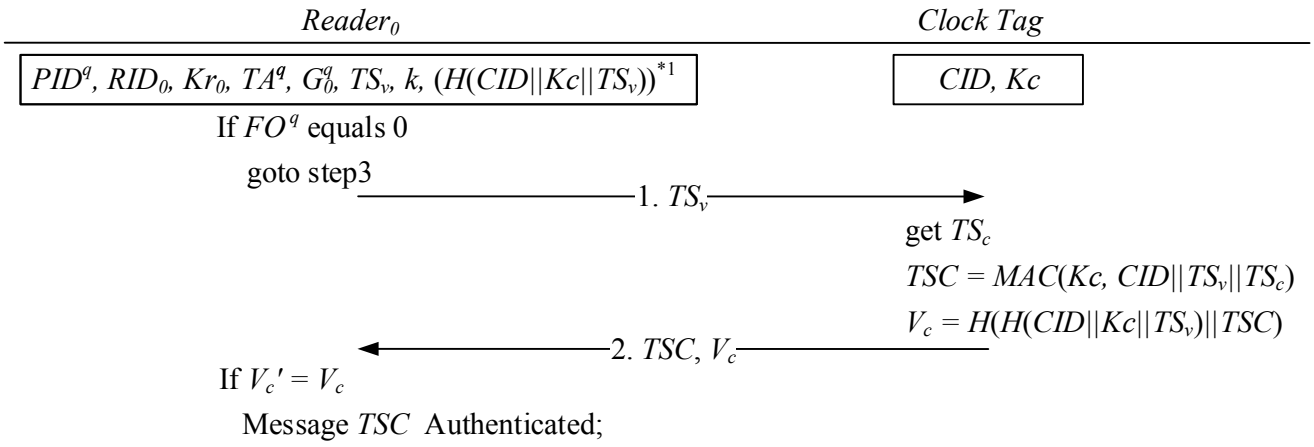


Figure 4. Read Tree with the reading capacity of six tags: (**a**) A single group key; (**b**) several group keys.

After group keys were distributed, the following explains the three stages regarding grouping proof protocol OMRGP proposed in this study: how can RID_0 generates a grouping proof under offline conditions in which the reader cannot be instantly connected to the backend verification server. From Figures 5–7, the contents in the boxes at the top indicate that the contents were information already written into protocol during the initial setting and before the execution of the protocol. First, through third-party active clock tag, Stage 1 obtained the trusted start timestamp for the system. Second, Stage 2 gradually generated pieces of grouping proof and inspection pieces from the read tree's leaf nodes, with the parent node combining all pieces of proof from child nodes until the tree root. Finally, Stage 3 incorporated affirming whether the grouping proofs signed by both the transporter and recipient and sent to the clock tag-grouping proof in the beginning was completed in time.

As illustrated in Figure 5, Stage 1 indicates that when transporter's reader RID_0 cannot connect to the verification server, the timestamp TS_v written into the reader in the initial setting must be first transmitted to trusted clock tag to acquire the trustworthy initial time. After the clock tag receives the TS_v transmitted by the reader RID_0, it uses the identification code *CID*, verifier's shared key K_C, tag's current timestamp TS_c, and the received timestamp TS_v to compute the signed timestamp *TSC*. The message verification code $V_c = H(H(CID||K_c||TS_v)||TSC)$ is also computed and transmitted to the reader RID_0 along with *TSC*. Finally, to authenticate the source of the received message *TSC*, the reader RID_0 uses the clock tag check code $H(CID||K_c||TS_v)$ received from the verifier and the acquired *TSC* to compute the message verification code V_c.



*1 When $FO^q = 1$, the clock tag message must be authenticated.

Figure 5. Signed timestamp acquired from trustworthy clock tag.

At Stage 2, the reader RID_0 uses the keys in the layered read-tree and child node (Figure 4) to encrypt recipient's identification code PID^q, timestamp TS_v, signed timestamp TSC, group key set RG_j^q for the child node reader RID_j^q, and tag verification code set RT_j^q and transmits them to all child nodes until all leaf nodes have been reached. For example, as indicated in Figure 4a, the reader RID_0 first uses the key SK_1^q to encrypt PID^q, TS_v, TSC, $RT_1^q = \{TH_{13}^q, TH_{14}^q, TH_{15}^q, TH_{16}^q, TH_{17}^q, TH_{18}^q\}$, and $RG_1^q = \{GK_8^q, GK_9^q\}$, and then transmits them to the reader RID_1^q, which employs the session key SK_1^q to decrypt the message and encrypt PID^q, TS_v, TSC, $RT_7^q = \{TH_{13}^q, TH_{14}^q, TH_{15}^q\}$, $RG_7^q = \{GK_8^q\}$, PID^q, TS_v, TSC, $RT_8^q = \{TH_{16}^q, TH_{17}^q, TH_{18}^q\}$, and $RG_8^q = \{GK_9^q\}$ by using the session keys SK_7^q and SK_8^q, and

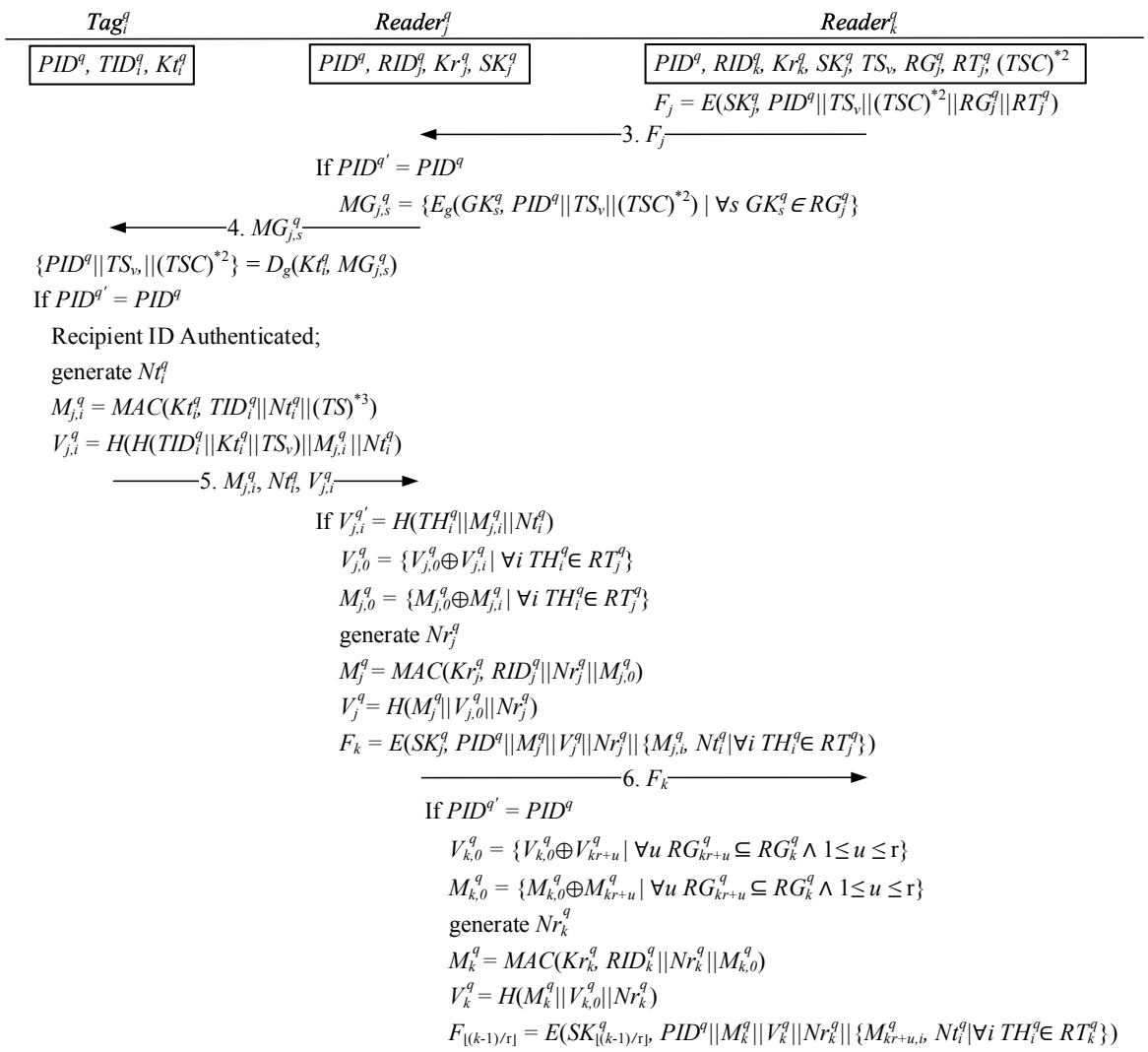
then sending them to the leaf node readers RID_7^q and RID_8^q. Subsequently, all leaf node readers collect the pieces of proof from the corresponding tags to generate a grouping proof, which is then transmitted to the upper levels and transmitted back to the reader RID_0.

Since the activities of any two readers in the read-tree are identical, the algorithm uses any leaf node reader RID_j^q in the read-tree receiving a request from the parent node reader RID_k^q to generate a grouping proof, as shown in the steps in Figure 6. After the leaf node reader RID_j^q uses its session key SK_j^q to decrypt a message F_j transmitted from the reader RID_k^q of the recipient PID^q, all group keys in RG_j^q are extracted and the multicast message $MG_{j,s}^q$ is encrypted using the keys PID^q, TS_v, and TSC and transmitted to each tag to generate the pieces of proof.

When any tag TID_i^q receives a multicast message $MG_{j,s}^q$ that is decrypted using the shared key Kt_i^q of the verifier, the decrypted messages are checked to determine whether they contain the correct PID^q in order to verify the read message. When the message is successfully verified, the shared key Kt_i^q of the verifier is employed to compute the pieces of proof $M_{j,i}^q$ assigned to a tag TID_i^q generated by RID_j^q for confirming the personal tag identification code TID_i^q, a randomly generated number Nt_i^q, and a timestamp (when the offline stamp and online stamp are TSC and TS_v, respectively). Subsequently, the hash value $H(TID_i^q||Kt_i^q||TS_v)$ for TID_i^q, Kt_i^q, and TS_v are computed together with the pieces of proof $M_{j,i}^q$ and a random number $Nt_{j,i}^q$ to generate the message verification code $V_{j,i}^q$ for the reader to reconfirm.

When the leaf node reader RID_j^q receives response messages from the tags, the obtained Nt_i^q, $M_{j,i}^q$, and tag verification value $TH_i^q = H(TID_i^q||Kt_i^q||TS_v)$ transmitted from RID_k^q (as indicated in Step 3) are computed to obtain $V_{j,i}^q$; in addition, the message verification code $V_{j,i}^{q\prime}$ transmitted by the tags are, this time, employed to inspect the message integrity and verify which tags transmitted the messages, which is derived from the other group members, in order to prevent malicious users from exploiting any of the recipients' tags that are not associated with this delivery, thereby blocking transmission of the proof. Subsequently, the reader RID_j^q combines all of the pieces of proof $M_{j,i}^q$ and the verification code $V_{j,i}^q$ by incorporating the XOR operation of commutative law, both of which are generated by the group member tags, into pieces of proof $M_{j,0}^q$ without sequence and message verification code $V_{j,0}^q$. Through the shared key Kr_j^q of the verifier, the pieces of proof M_j^q generated by the reader are computed using the reader identification code RID_j^q and the randomly generated numbers Nr_j^q and $M_{j,0}^q$; along with $V_{j,0}^q$ and Nr_j^q, the message verification code V_j^q is generated for all of the tags. The session key SK_j^q is used to encrypt PID^q, M_j^q, V_j^q, Nr_j^q, $M_{j,i}^q$, and Nt_i^q for all group member tags, which are transmitted back to parent node reader RID_k^q.

After the parent node reader RID_k receives the response message F_k transmitted by child node reader RID_j^q, the session key SK_j^q is first used to encrypt the message to confirm that the message contains the same recipient PID^q in order to ensure the correctness of the message. Subsequently, using the same method as the leaf node reader, all of the received pieces of proof M_j^q and message verification codes V_j^q of the child node reader are used to generate the pieces of proof M_k^q for the reader, and the message verification code V_k^q is transmitted with message $F_{\lfloor (k-1)/r \rfloor}$ to the reader at the upper level.



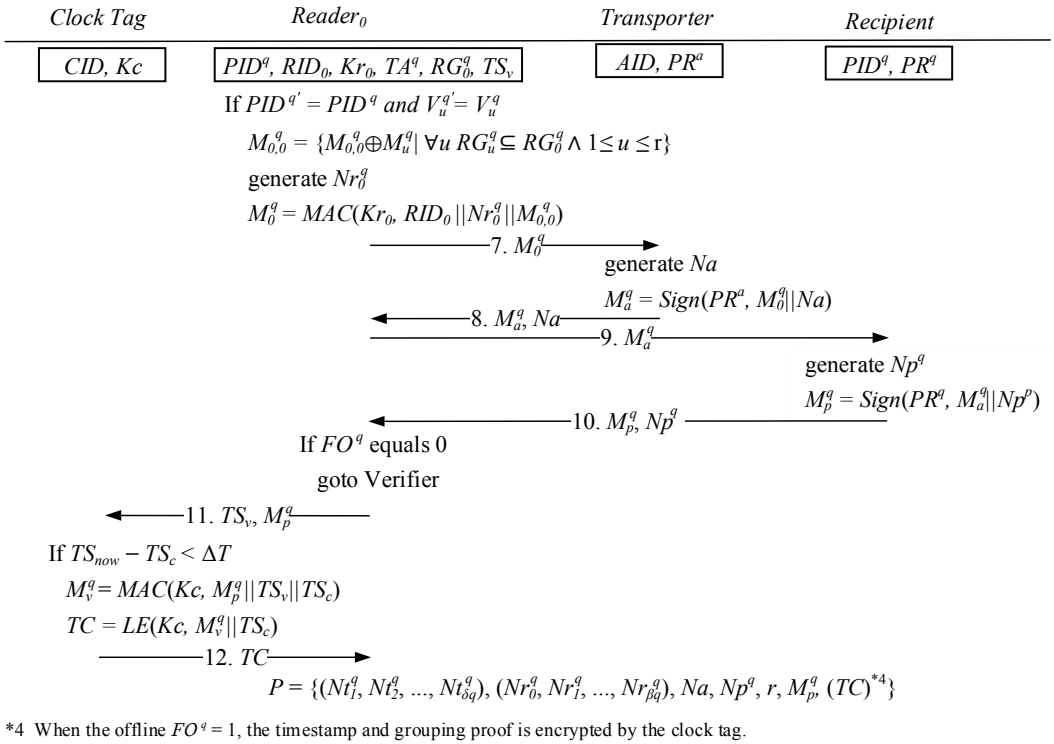
*2 When offline $FO^q = 1$, the timestamp is generated by the trusted clock tag.
*3 When offline $FO^q = 1$, $TS = TSC$; else, $TS = TS_v$.

Figure 6. Generating grouping proofs by multilayered reader.

As shown in Figure 7, Stage 3 indicates that after the reader RID_0 receives a message from the recipient PID^q, the tag verification value TA^q generated by the verification server, and the Nt_i^q, $M_{j,i}^q$, Nr_j^q, and M_j^q transmitted by the child node reader are computed to confirm whether they match the message verification code V_j^q to reconfirm the message integrity and verify the tag message. In addition, the shared key Kr_0 of the verifier is employed to generate the grouping proof M_0^q, which is then transmitted to the transporter's tags depending on the results obtained from the identification code RID_0 of the reader, and a randomly generated number Nr_0 and all of the excluded pieces of proof sent back from the child node.

When the transporter's tag receives the grouping proof M_0^q from the reader RID_0, a random number Na is generated, and the transporter's private key PR^a is used with the signing function to compute M_0^q and the signed proof M_a^q; subsequently, M_a^q and Na are transmitted back to the reader RID_0. After the reader RID_0 receives the message M_a^q signed by the transporter, M_a^q is then transmitted to the recipient's tag for signing. Using the randomly generated numbers Np^q and M_a^q and the private key PR^q, the recipient computes the signed proof M_P^q; finally, M_p^q and Np^q are then transmitted back to the reader RID_0.

If the reader RID_0 cannot connect to the verifier when the receipt proofs and pick proofs are signed by the transporter and recipient, then the timestamp TS_v and signed proof M_p^q must be transmitted back to trusted clock tag to verify that parts of the cargo could not have been moved to other locations while the proofs were being generated. When the clock tag receives a message indicating that the difference between the system time and the time when grouping proof was initialized (TS_c) is below the threshold value, the shared key Kc of the verifier can be employed to generate the final grouping proof M_v^q for M_p^q, TS_v, TS_c, thereby providing evidence that all of the tags, the transporter, and the recipient are in the same interval. Using a light symmetric key encryption method, the key Kc encrypts the grouping proofs M_v^q and TS_c into the message TC, which is transmitted back to the mobile reader and then to the verifier once a connection becomes available, for the protocol to be finalized. However, if the reader RID_0 can connect to the verifier when it receives the grouping proofs signed by the transporter and the recipient, then the clock tag is not needed and the grouping proof can be directly transmitted to the verifier to confirm whether the grouping proof has been completed within the time threshold.



*4 When the offline $FO^q = 1$, the timestamp and grouping proof is encrypted by the clock tag.

Figure 7. Affirm the tags and proofs signed by both sides and verify the time constraint.

2.3. Dispute Resolution Phase

When the transporter transmits the final grouping proof P to the verifier, the verifier confirms the integrity of the transporter, recipient, and cargo. First, when $FO^q = 0$, the shared key Kc of the clock tag is employed to decrypt in order to obtain the grouping proof M_v^q and start time TS_c in message TC; subsequently, the recipient's random number Np^q and public key PK^q are used to decrypt the grouping proof M_v^q in order to obtain M_a^q, and the transporter's random number Na and public key PK^a are employed to decrypt M_a^q in order to obtain the grouping proof M_0^q. According to the total number of tags δq, segments k, readers βq, and the maximum number of concurrent scans r, the read-tree and grouping

key are surmised from and adopted for generating the proofs. In accordance with the proposed protocol, M_0^q is recomputed and compared with the transmitted grouping proof $M_0^{q\prime}$ to determine whether the two are identical. Finally, according to the shared key Kc, start time TS_v, and timestamp TS_c of the clock tag, the message verification code $M_v^q = MAC\left(K_c, M_p^q||TS_v||TS_c\right)$ is computed to reconfirm whether the code corresponds with the authentication message provided by the clock tag, thus effectively completing the grouping proof. By contrast, when $FO^q = 0$, the verifier first computes whether the time difference between the current system time and timestamp TS_v is below the threshold value; subsequently, M_0^q is computed by the read-tree, and M_v^q is confirmed to complete the reinspection through the proofs signed by both parties and the computed random number $M_0^{q\prime}$.

Therefore, in the process of generating the grouping proofs, the transporter and recipient each verify all of the involved tags and use their personal private keys to sign the proofs; thus, the grouping proofs guarantee the rights of both parties. Specifically, when cargo is received by the recipient, the transporter has undeniable proof that the recipient has received the cargo. In addition, if the recipient needs to return cargo through the transporter, the same protocol can be applied, except that the roles of recipient and transporter are swapped. The recipient also has the signed proof indicating that the cargo has been returned to the transporter, thus preventing the transporter from denying that cargo has been retrieved.

3. Security and Performance Analysis

The proposed method and the Internet connection method for the verifier use secure frameworks and can therefore be trusted. Extant mechanisms can also be employed so that the transporter's reader RID_0 can extract the recipient's identification code PID^q, timestamp TS_v, tag check code TA^q, and group of keys G_0^q by using the current security verification procedure to ensure that the connection is secure before the protocol proceeds. The following analysis is primarily focused on determining whether the protocol proposed in this study can prevent most known malicious attacks aimed at exploiting grouping proofs transmitted between tags and readers, and whether it can guarantee anonymity and message integrity.

Prevent Replay Attack

Malicious users intercept a message containing a previously generated grouping proof through an eavesdropping, and the previously captured message is replayed to produce grouping proofs for nonexistent tags. However, because any piece of proof for the tags incorporates a random number Nt_i^q, timestamp TSC, or TS_v generated by tags, the reader can thus detect errors and ignore the replay message by using the message check code transmitted from the verifier to authenticate the received message.

Prevent Tag Impersonation

The tags generate the pieces of proofs $MAC(Kt_i^q, TID_i^q||Nt_i^q||TS)$ and a message verification code $V_{j,i}^q$ from the shared key Kt_i^q of the verifier, and the reader stores the message verification code required for reading the tags, which is generated by the verifier. Because malicious users do not have the required shared key Kt_i^q of any tag in the tag group, the pieces of proofs generated by impersonated tags cannot pass the reader's or the verifier's authentication process.

Prevent Multi-session Attack

If multiple readers simultaneously generate grouping proofs, the leaf node reader stores the tag check codes of all of the members in this tag group; thus, tags that are not assigned to the group cannot pass the authentication process; and thus, tag impersonation attacks are ineffective. Consequently, malicious users cannot forge extra grouping proofs by crisscrossing pieces of proofs derived from two different groups.

Prevent Denial of Proof

In addition to generating pieces of proof for every tag, the protocol also generates the message verification code $VA_{j,i}^{q}$ with the tag verification code $H(TID_i^q||Kt_i^q||TS_v)$. Although the leaf node reader has no shared key Kt_i^q for the tags and cannot generate tag verification codes, using the cargo integrity check code $TH_i^q = H\left(TID_i^q||Kt_i^q||TS_v\right)$ provided by the verifier, it can confirm whether a response message has cargo tag members that do not belong to this delivery, but are instead generated by a malicious user. Therefore, the condition of authentication failure being generated by the verifier despite the existence of all legitimate tags is prevented.

Prevent Concurrency Attack

When two readers simultaneously use the same tags, parameter confusion can occur, which enables a malicious reader to scan tags by crisscrossing tags, and block grouping proofs. However, in the proposed protocol, no cargo tag has a temporary parameter, and the reader needs to communicate with the tags only once to generate the pieces of proof. Therefore, it is impossible for a concurrency attack to occur.

Anonymity

In the proposed protocol, all messages used by the reader are multicast messages that do not contain specific tag information. In addition, the pieces of proof M_i^q and message check code $V_{j,i}^q$ transmitted by any tag TID_i^q are computed from using hash functions, along with the random numbers Nt_i^q generated each time and a shared key Kt_i^q; thus, the anonymity of the cargo tags can be protected. In the final signed message, the confidentiality of the transporter's and recipient's tags is protected by random numbers Na and Np^q, respectively.

Prevent Tracking Attack

The protocol proposed in this study can protect the anonymity of all the involved tags. The messages transmitted by a tag change according to the random number Nt_i^q, which is generated each time a message is sent, and the reader also uses a different session key for every message. Therefore, the relationship among the messages containing proofs for any tag cannot be obtained by analyzing multiple grouping proofs; thus, the protocol ensures the confidentiality of the location of the cargo tags to prevent the cargo from being tracked by malicious users.

Message Integrity

The pieces of proof $M_{j,i}^q$ transmitted back to the reader by the tags are require a random number Nt_i^q in order to be calculated; thus, when a malicious user intercepts the random numbers, despite the pieces of proof being generated by legitimate tags, the proofs cannot be successfully reconstructed by the verifier because the random numbers are different for every message. Therefore, the message verification

code $V_{j,i}^{q}$ is employed to ensure that a response message has not been modified in order to ensure the message integrity.

Table 2 shows the OMRGP proposed in this study and other grouping proof methods to compare whether they can protect against the major types of attacks targeting grouping proofs: replay attack, tag impersonation attack, multisession attack, denial of proof, concurrency attack, and tracing attack. The protocols can protect against those marked with an "O"; those marked with an "X" are a security threat; and those marked with "Δ" can be are not a threat so long as certain conditions are satisfied.

Table 2. Comparison table indicating the security of grouping proof.

Protocol	Replay Attack	Tag Impersonation	Multi-Session Attack	Concurrency Attack	Denial of Proof
Burmester *et al.* [10]	O	O	O	X	Δ^2
Saito *et al.* [13]	X	X	O	X	X
Lin *et al.* [18]	O	O	O	O	X
Sun *et al.* [19]	O	O	O	Δ^1	X
Hermans *et al.* [20]	O	O	O	X	X
Lo *et al.* [21]	O	O	O	O	X
Ma *et al.* [22]	O	O	O	O	X
Chien *et al.* [24]	O	O	O	X	X
Peris-Lopez *et al.* [26]	O	O	O	O	X
Piramuthu [27]	O	O	X	X	X
Sundaresan *et al.* [28]	O	O	O	O	O
Yen *et al.* [32]	O	O	O	O	Δ^2
Leng *et al.* [37]	O	O	O	X	X
Huang *et al.* [44]	O	X	O	X	X
OMRGP	O	O	O	O	O

Note: Δ^1: Not overwriting the proofs from different readers; but the random numbers generated by proofs may still be overwritten; Δ^2: Filters proofs that do not belong to a group of tags, but cannot prevent a denial of proof attack because of the compromised proof integrity.

Table 2 shows that the grouping proof proposed in this study can protect against all major attacks currently in use. In the method proposed by Saito *et al.* [13], the tags generate messages but do not use random numbers that change for every message; consequently, malicious users can generate counterfeit tags by replaying old messages to generate grouping proofs [18,27,31]. The method proposed by Huang and Ku [44] can be exploited by replacing parts of the pieces of proof to forge tags [26,45] and authentication can be avoided if the verifier has listed tag as redundant in its cyclic redundancy check. Peris-Lopez *et al.* [15] showed that the method proposed by Piramuthu [27] was flawed because it enables malicious users to eavesdrop and intercept pieces of grouping proofs by crisscrossing two identical time intervals to forge an additional third proof. In addition, according to the methods proposed by Saito *et al.* [13] and Piramuthu [27], tags are read and written multiple times to generate grouping proofs; this causes the problem in which the previously written content can be overwritten by other readers [21]. Moreover, various methods for generating grouping proofs [10,20,24,37,44] require the reader to read the tags more than twice when generating grouping proofs. However, the tags cannot verify readers; thus, when several readers simultaneously generate grouping proofs for the same tag, concurrency attacks can arise, which can prevent grouping proofs from being generated because the

contents in previous tags are overwritten by subsequent readers. The method proposed by Sun *et al.* [19] requires the read group to be inspected every time tags generate proofs; thus, the proofs read by different readers are not overwritten; however, the random numbers are not subjected to the same security check in the previous step and may therefore be overwritten. Because the proposed method can complete the grouping proofs by reading and writing on the tags and because random numbers are also used in addition to the identification codes used in the prior authentication process, the proposed protocol prevents erroneous grouping proofs from being generated, thereby protecting against replay attacks, tag impersonation, multisession attacks, and concurrency attacks.

In addition to the correctness of the generated proofs, the readers do not authenticate the response message tags in the grouping proof methods of [13,18–22,24,26,27,37,44]; thus, if response messages generated from tags that do not belong to the specific group of tags are included, the verifier rejects the messages and discards the proofs, resulting in a denial of proof [28]. Yen *et al.* [32] and Burmester *et al.* [10] have proposed that message integrity must be authenticated to prevent parts of a message from being modified, which causes the problem in which legitimate proofs cannot be authenticated by the verifier, resulting in a denial of proof [28]. Therefore, in the present study, readers were employed to verify all of the collected tags to prevent including tags that do not belong to the group and to avoid denial of proof from occurring. Finally, the method proposed by Sundaresan *et al.* [28] can protect known attacks on grouping proofs. However, as shown in Table 3, because the proposed method cannot enable all of the involved proof tags to reduce the time for generating grouping proofs through parallel computing, therefore, attackers can exploit this time difference to generate legitimate grouping proofs from a group of tags that do not exist in the same time and location [36]; consequently, this method is inapplicable to SCM where large volumes of cargo are involved.

Table 3. Comparison of grouping proof performance.

Protocol	Anonymity	Tracking Attack	Offline	Order Independent	Simultaneity
Burmester *et al.* [10]	O	O	O	X	Δ^4
Saito *et al.* [13]	O	Δ^3	X	X	X
Lin *et al.* [18]	X	X	O	X	X
Sun *et al.* [19]	O	O	O	O	O
Hermans *et al.* [20]	O	O	O	O	O
Lo *et al.* [21]	O	O	O	X	X
Ma *et al.* [22]	O	O	O	X	X
Chien *et al.* [24]	O	Δ^3	X	X	X
Peris-Lopez *et al*. [26]	O	O	X	X	X
Piramuthu [27]	X	X	X	X	X
Sundaresan *et al*. [28]	O	O	O	X	X
Yen *et al.* [32]	O	O	X	O	O
Leng *et al.* [37]	X	X	X	O	X
Huang *et al.* [44]	X	X	O	X	X
OMRGP	O	O	O	O	O

Note: Δ^3: Single message that features anonymity; however, relevance among tags with messages from different sessions can be used to track tag movement; Δ^4: Only parts of the tags in the group can concurrently compute pieces of proof.

Table 3 indicates that the proposed system can prevent cargo from being tracked and provide anonymity while operating in the offline phase; simultaneously, pieces of proof that do not need to follow the tag sequence can be generated to achieve the five types of security characteristics involved in processing grouping proofs. However, to ensure confidentiality, the system must be designed to prevent malicious attackers from obtaining the identity of the tag owner by eavesdropping on grouping proofs. Because many grouping proof methods [18,27,37,44] involve directly transmitting identification codes to the readers without first anonymizing the tag owner's identity; consequently, the cargo owners' private information can be stolen and the location of their cargo can be tracked [28]. Moreover, although the methods proposed by Chien *et al.* [24] and Saito *et al.* [13] ensure tag anonymity and thus ensure that malicious attackers cannot track the tagged cargo simply by monitoring the identification codes of the tags, nevertheless, because the request acknowledgement response messages sent by the tags are identical, malicious attackers can track the location of the tagged cargo by eavesdropping on multiple messages [26]. Therefore, this study incorporated random numbers into the messages to scramble the responses for preventing from being tracked in the supply chain, thereby achieving location privacy.

According to the method proposed by Peris-Lopez *et al.* [26], a reader must be able to connect to the verifier for it to obtain a timestamp of when the proofs were generated; subsequently, the grouping proofs are immediately sent to the verifier to compare the time [22]. Similar grouping proof methods [13,24,27,32,37] also require immediate authentication from the verifier, and are unsuitable for generating grouping proofs in the offline phase.

The grouping proofs proposed by Saito *et al.* [13] and Piramuthu *et al.* [27] pertain to conventional proofs generated by all tags on site one after another; thus, the verifier must verify the tags in the order that they were generated [29]; furthermore, methods for generating grouping proofs [10,18,21,22,24,26,44] typically have a particular sequence. Because the method proposed by Leng *et al.* [37] involves unicasting messages to tags, the participating members cannot concurrently conduct computation [19]. In the method proposed by Burmester *et al.* [10], the tree structure can permit only a few tags in the group to concurrently compute the pieces of proof, and the reader must follow the predetermined sequence when collecting the proofs. Therefore, the present study adopted the multicasting method to simultaneously generate the pieces of proof for all tags and can collect the grouping proofs without adhering to any sequence through the XOR operation of commutative law.

4. Effectiveness Analysis

Because the grouping proofs generated in sequence require the time complexity O(m!) when being authenticated by the verifier, this section compares the proposed OMRGP method only with those grouping proof methods that do not require a predetermined sequence [19,20,32,37] to examine the computing and transmission time for the tag and reader to generate proofs. To ensure that the comparison is objective, the experiments were conducted under the following constraints: each method involved using a reader that can scan r tags [39] to generate grouping proofs for m tags [46] at a rate of 3.55 M clock cycles per second. In addition, an error-correcting code and asymmetric encryption function with the same security strength (2^{80} bits) were employed.

Specifically, T_{SE} denotes the computation time for conducting symmetric encryption and decryption [47], T_{EC} indicates the time for conducting elliptic curve encryption and decryption [48], T_G

represents the time for encrypting and decrypting a group key [49], T_{RNG} denotes the required time for generating a random number [50], T_H is the computation time for executing a hash function [47], and T_{SIG} indicates the required time for proof signing [51]. In addition, because XOR logic operation can be neglected compared to the aforementioned computation time, the formulas in Table 4 do not consider the required time for this type of operand. To simplify the comparison, the computing capacity of the reader was adopted to present the required computation time for devices with a powerful arithmetic capability, as demonstrated by the additional timestamps used in the various methods.

Table 4. Computational capacity of grouping proof tags (with *m* number of tags).

Name of the Method	Cargo Tag	Mobile Reader
Sun *et al.* [19]	$\lceil m/r \rceil (2T_{SE} + T_{RNG})$	$T_{SE} + 2T_H$
Hermans *et al.* [20]	$\lceil m/r \rceil (2T_{EC} + 2T_{RNG})$	$T_{SIG} + T_{RNG}$
Yen *et al.* [32]	$\lceil m/r \rceil (7T_{RNG})$	$2T_{SIG} + \mathrm{m}(T_{RNG}) + 5T_{RNG}$
Leng *et al.* [37]	$m(2T_H + T_{RNG})$	$m(T_H) + \frac{m(T_H)}{r} + T_H$
OMRGP	$T_G + 3T_H + T_{RNG}$	$T_G + 2T_{SIG} + 3T_{SE} + 7T_H + 3T_{RNG}$ $(\lceil \log_r(m/r) \rceil)(2T_{SE} + 2T_H + T_{RNG})$

Table 4 indicates the computational capacity of *m* tags according to the grouping proofs generated by the reader with *r* capacity for the maximum number of tags that can be scanned concurrently. Therefore, for the OMRGP method proposed in this study, each reader can manage a maximum of *r* and thus only one multicast is to be broadcasted to all tags. The grouping proof methods in [19,20,32] also send multicast messages to all tags; however, when $m > r$, the reader must transmit the message multiple times; thus, *m* tags required a computation time of $\lceil m/r \rceil$ times. According to the method proposed by Leng *et al.* [37], a reader must send different messages to each tag, and each tag requires its own computation, in that *m* tags ultimately require *m* times of computation time. By contrast, Leng *et al.* indicated that the reader should assign messages to each tag; Yen *et al.* verified the identification code for individual tags, in which the computational capacity of the reader increased with the number of tags; and Hermans *et al.* and Sun *et al.* have employed methods in which identical messages were broadcast to all tags; thus, the required computational capacity for the reader to generate grouping proofs remained constant. The proposed method was designed for operation in a multilayered reader; despite a similar message is broadcast, the readers are required to communicate with other readers, thereby increasing the computational capacity to $\log_r m/r$ times. In Figures 8 and 9, readers with a maximum reading capacity of 200 tags were employed to analyze the computing time required by various methods when the number of tags and group tags in the reader doubled from 100 each time until the quantity reached 12,800. Since we use a group key with the tree height of 2, a reader can multicast message to 200 tags within the capacity that can be read by the reader. In the following simulations, 100 tags and 200 tags need two readers, 400 tags need three readers, 800 tags need five readers, 1600 tags need nine readers, 3200 tags need 17 readers, 6400 tags need 33 readers, and 12,800 tags need 65 readers.

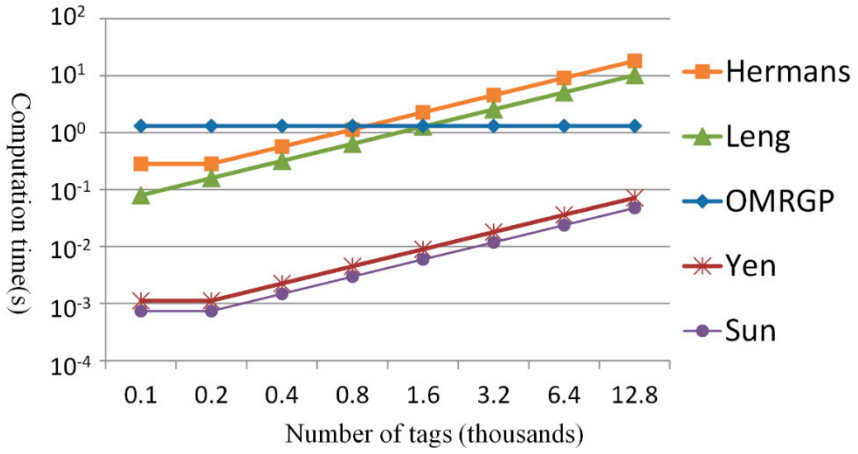


Figure 8. Comparing the computational load of the tags.

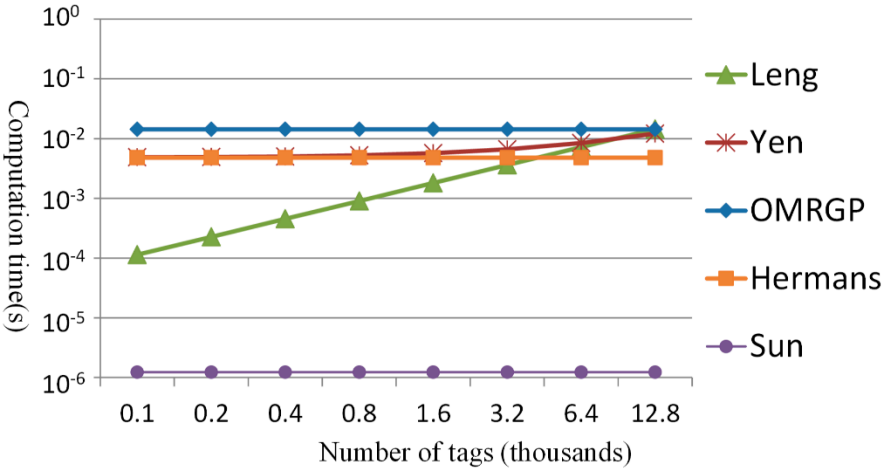


Figure 9. Comparing the computational load of readers.

According to Figure 8, when the proposed OMRGP method involved fewer than 200 tags, more computing time was required because the group key has to be decrypted. When the number of tags exceeded 200, the number of grouping proofs [19,20,32,37] increased with the number of tags; thus, the computing time and tag number were linearly related. Therefore, when the grouping proofs were generated for an extreme number of tags, the tag computational capacity of the proposed OMRGP method was more efficient compared with the other methods. In addition, Figure 9 indicates the computational capacity of the reader in generating grouping proofs. Although the same message was broadcast, the required computation time was more than that of the other multicast methods [19,20,32] because the group messages must be encrypted, messages must be transmitted between readers, and proofs must be signed by both the transporter and the recipient. Because the method proposed by Leng *et al.* [37] adopted a unicast method, the required computation time under conditions involving extreme number of tags was higher than that required from using OMRGP method.

This study subsequently compared the required transmission time for the proposed OMRGP with that of the other methods, for which L_{ID} denotes the length of a tag identification code (based on ISO-18000-6), L_{SE} is the message length after applying symmetric encryption, L_{EC} indicates the message length after applying elliptic curve encryption, L_G represents the message length after performing group key

encryption, L_{RNG} indicates the message length for a random number, L_H represents the message length of a hash function, and L_{SIG} represents the required message length after signing the proof.

Table 5 indicates the transmission capacity of the grouping proofs generated by *m* tags. Because the proposed OMRGP method adopted a multilayered grouping proof structure, a maximum or *r* tags were distributed to each reader; thus, compared with the other methods, increasing the cargo volume did not increase the transmission time from the tags to the verifier. Moreover, in the transmission from the reader to tags, a read-tree was employed; consequently, the transmission time between readers increased $\lceil \log_r(m/r) \rceil$ times. In the methods proposed by Hermans *et al.*, Yen *et al.*, and Sun *et al.*, the readers could not manage *m* tags simultaneously; consequently transmissions were repeated $\lceil m/r \rceil$ times. According to the subgrouping proof of Leng *et al.*, the reader transmitted the message $\lceil m/r \rceil + m$ times. For the sake of objectivity, all methods adopted the electronic product code Class-1 Generation 2 (EPC Class-1 Gen2), with the network bandwidth of the tags and readers set to 160 and 640 kbps, respectively [38]. The message lengths for L_{ID}, L_{SE}, L_{RNG}, and L_H were arbitrarily set at 64 bits, and the message lengths for L_{ECC} and L_G were arbitrarily set at 192 and 1024 bits, respectively.

Table 5. Transmission capacity of *m* grouping proof tags.

Name of the Method	From Tag to Reader	From Reader to Tag (or Reader)
Sun *et al.* [19]	$m(L_{ID} + 2L_{SE})$	$\lceil m/r \rceil (3L_H)$
Hermans *et al.* [20]	$m(2L_{EC} + L_{RNG})$	$\lceil m/r \rceil (L_{RNG})$
Yen *et al.* [32]	$m(4L_{RNG})$	$\lceil m/r \rceil (3L_{RNG})$
Leng *et al.* [37]	$m(2L_{ID} + L_H + L_{RNG})$	$\lceil m/r \rceil (2L_{ID} + L_{RNG}) + m(L_{ID} + L_H)$
OMRGP	$r(2L_H + L_{RNG})$	$L_G + (\lceil \log_r(m/r) \rceil)(2L_{SE})$

Figures 10 and 11 show the transmission time for the tags to generate the grouping proofs. The methods of Sun *et al.* and the one proposed in the present study transmitted messages that were identical in length ($3 \times 64 = 192$ bits); however, the proposed method divided all of the tags into several groups and every group could process concurrently. When the number of tags exceeded 200, the time for the reader to collect the tags, according to Sun *et al.*, exceeded the fixed transmission time suggested in the proposed OMRGP method. In addition to the method proposed by Sun *et al.*, the other three methods had to transmit messages that were longer than 192 bits. Hence, when >200 tags were involved, the transmission time of the proposed method was shorter than that of the other methods. Figure 11 depicts the time required for the reader to transmit messages. Leng *et al.* [37] did not adopt a multicasting protocol for transmitting messages, which increased the reader's transmission time with an increased number of tags. When more than 200 tags were involved, the grouping proof methods of [19,20,32] were not able to scan all tags in one shot because of the reading capacity of the readers, which divided and read for several cycles, thereby increasing the transmission time. In this present study, grouping proofs were generated using the group key and the multilayered read-tree. When there was only a few tags involved, the proposed OMRGP method took longer to read than that other multicast methods; however, an observation can be made that the OMRGP method is more efficient compared with the other methods when a considerable number of tags is involved.

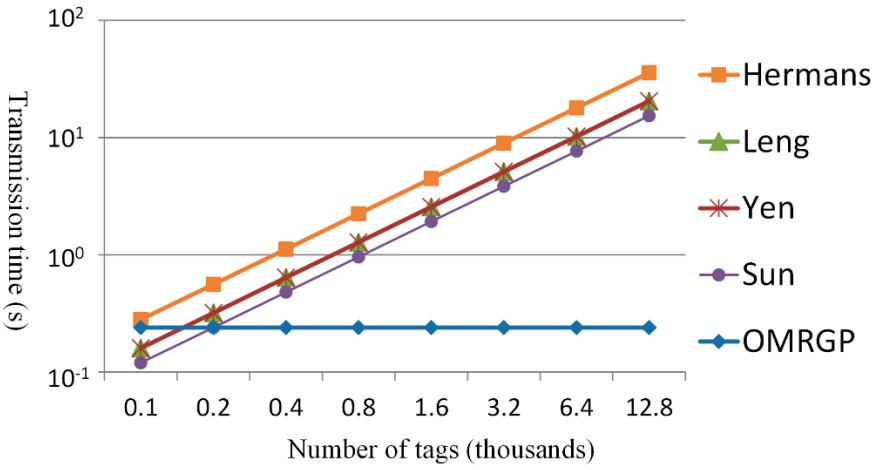


Figure 10. Comparing the message quantity of the collected tags.

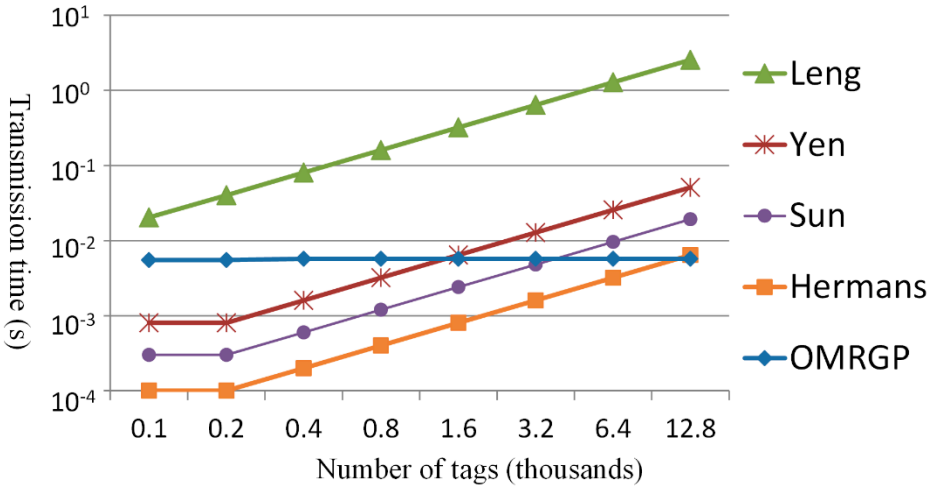


Figure 11. Comparing the message quantity transmitted by the reader.

Figure 12 shows the total amount of time consumed when several tags generated the grouping proofs, including the computing and transmission times of the tags and readers involved. In the OMRGP method, although an increased time was taken for computation, the transmission capacity was evidently larger than the computational capacity and thus prevented the time for generating grouping proof to increase considerably with an increased number of tags. In addition, the OMRGP method features a mechanism for verifying tags. According to Sun *et al.*, readers that were not required to verify tags reduced computation load; however, the time for generating grouping proofs when an extreme number of tags were involved exceeded the time for the OMRGP method. Yen *et al.* employed a random function to generate grouping proofs; groups could not be selected under their method; thus, under the condition in which a mechanism for verifying tags existed, the effectiveness of OMRGP method becomes more obvious once the number of tags exceeds a certain threshold. Leng *et al.* adopted a unicast method, in which the reader was required to perform a high number of transmissions; therefore, in less time than it would take unicast grouping proofs, the OMRGP method can generate grouping proof in advance and when there are fewer tags. Finally, Hermans *et al.* adopted a high elliptic curve encryption and decryption to generate grouping proofs, demonstrating the effectiveness of the OMRGP method when there are fewer tags.

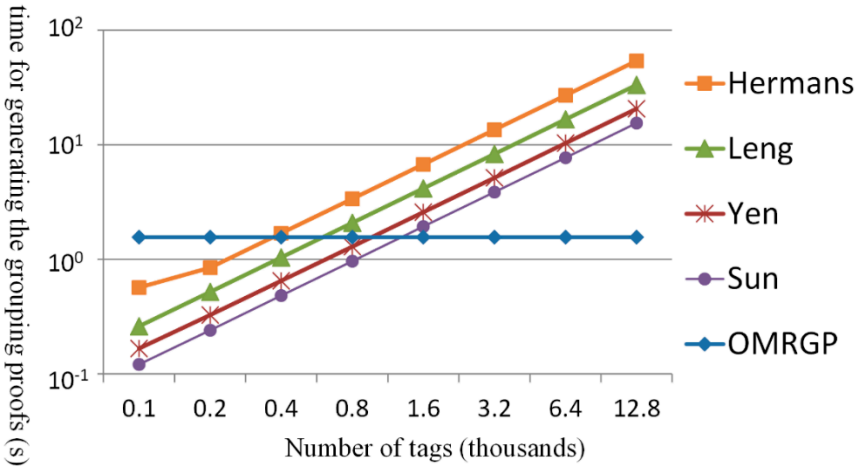


Figure 12. Comparing the time for generating grouping proofs.

5. Conclusions

This study proposed a method for generating multilayered grouping proofs to solve the disputes over the loss of cargo when high-quantity shipments are transferred in the supply chain. Through the layered parallel scans, the requirement in which the generated grouping proofs must be read in batches because of the maximum tag-reading capacity constraint on a reader in the supply chain environment can be solved. Group keys were employed to distribute the tags corresponding to each reader to ensure that the tags are not repeatedly read, thus exceeding the time threshold. In addition, both the transporter and recipient were allowed to verify the cargo and sign the proofs to guarantee the integrity of the grouping proof. The anonymity and message integrity characteristics of the OMRGP method can defend against most of the currently known attacks on grouping proofs: replay attack, multi-session attack, tag impersonation attack, denial of proof, and tracing attack. The OMRGP method overcame the problem of at least one type of characteristic not complying with the security standards, a problem possessed by most studies. This study also analyzed the computation load of the tag and reader. The effectiveness of grouping proof protocols were compared, and the results show that when an extreme number of tags are involved, the increase in the number of tags did not evidently increase the time for generating grouping proofs under the proposed protocol. Consequently, the protocol can be applied to SCM to reduce the time required to generate grouping proofs, and prevent exceeding the time threshold value for generating grouping proofs, thus preventing attackers from hijacking tags when the grouping proof is being processed, causing grouping proof problems.

Acknowledgments

The authors would like to express the sincere thanks to the reviewers for their invaluable comments and suggestions. This work is supported by the Ministry of Science and Technology, Taiwan, under Grant Nos. MOST 104-2221-E-033-020.

Author Contributions

All of the authors contributed extensively to the work. The original idea was conceived by Ming Hour Yang and Jia Ning Luo. Ming Hour Yang is the main author; he developed the methodology, supervised the research and revised the manuscript. Jia Ning Luo contributed to the manuscript's discussion and edition. Shao Yong Lu contributed with data processing and results analysis. The authors approved the final manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Engberg, S.J.; Harning, M.B.; Jensen, C.D. Zero-knowledge Device Authentication: Privacy & Security Enhanced RFID preserving Business Value and Consumer Convenience. In Proceedings of the 2nd Annual Conference on Privacy, Security and Trust, Moncton, New Brunswick, Canada, 13–15 October 2004; pp. 89–101.
2. Pisarsky, G.M. RFID Technology: An Analysis of Privacy and Security Issues. In Proceedings of the Computer Science Seminar SA3-T1–1, Hartford, CT, USA, 24 April 2004; pp. 1–5.
3. Koh, C.E.; Kim, H.J.; Kim, E.Y. The impact of RFID in retail industry:Issues and critical success factors. *Shopp. Cent. Res*. **2006**, *13*, 101–117.
4. Michael, K.; McCathie, L. The Pros and Cons of RFID in Supply Chain Management. In Proceedings of the International Conference on Mobile Business, Sydney, Australia, 11–13 July 2005; pp. 623–629.
5. Ekwall, D.; Lantz, B. Seasonality of cargo theft at transport chain locations. *Int. J. Phys. Distrib. Logist. Manag*. **2013**, *43*, 728–746.
6. Ekwall, D.; Lantz, B. Cargo theft at non-secure parking locations. *Int. J. Retail Distrib. Manag*. **2015**, *43*, 204–220.
7. Ray, B.R.; Chowdhury, M.; Abawajy, J. Critical Analysis and Comparative Study of Security for Networked RFID Systems. In Proceedings of the 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Honolulu, HI, USA, 1–3 July 2013; pp. 197–202.
8. Liang, Z.; Rodrigues, J.J.P.C. Service-oriented middleware for smart grid: principle, infrastructure, and application. *IEEE Commun. Mag.* **2013**, *51*, 84–89.
9. Lo, C.; Hsieh, W.; Huang, L. The Implementation of an Intelligent Logistics Tracking System Utilizing RFID. In Proceedings of the International Conference on Electronic Business, Beijing, China, 18–20 October 2004; pp. 199–204.
10. Burmester, M.; Munilla, J. Group-Scanning for RFID Supply Management. In Proceedings of the IEEE RFID Technology and Applications Conference, Tampere, Finland, 8–9 September 2014; pp. 266–271.
11. Lien, Y.H.; His, C.T.; Leng, X.; Chiu, J.H.; Chang, H.K. An RFID basedMulti-batch supply chain systems. *Wirel. Pers. Commun.* **2012**, *63*, 393–413.

12. Zhong, Z.; Qiu-Liang, X. Universal composable grouping-proof protocol for RFID tags in the internet of things. *Chin. J. Comput.* **2011**, *34*, 1188–1194. (In Chinese)
13. Saito J.; Sakurai, K. Grouping Proof for RFID Tags. In Proceedings of the International Conference on Advanced Information Networking and Applications, Taipei, Taiwan, 28–30 March 2005; pp. 621–624.
14. Juels, A. "Yoking-Proofs" for RFID Tags. In Proceedings of the 2nd Annual International Conference on Pervasive Computing and Communications, Orlando, FL, USA, 14–17 March 2004; pp. 138–143.
15. Peris-Lopez, P.; Hernandez-Castro, J.C.; Estevez-Tapiador, J.M.; Ribagorda, A. Solving the Simultaneous Scanning Problem Anonymously: Clumping Proofs for RFID Tags. In the Proceedings of the 3rd International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, Istanbul, Turkey, 19 July 2007; pp. 55–60.
16. Peris-Lopez, P.; Safkhani, M.; Bagheri, N.; Naderi, M. RFID in eHealth: How to combat medication errors and strengthen patient safety. *J. Med. Biol. Eng*. **2013**, *33*, 363–372.
17. Yu, Y.C.; Hou, T.W.; Chiang, T.C. Low cost RFID real lightweight binding proof protocol for medication errors and patient safety. *J. Med. Syst*. **2012**, *36*, 823–828.
18. Lin, C.C.; Lai, Y.C.; Tygar, J.D.; Yang, C.K.; Chiang, C.L. Coexistence Proof Using Chain of Timestamps for Multiple RFID Tags. In Proceedings of the Web and Network Technologies and Information Management, Huang Shan, China, 16–18 June 2007; pp. 634–643.
19. Sun, H.M.; Ting, W.C.; Chang, S.Y. Offlined Simultaneous Grouping Proof for RFID Tags. In Proceedings of the 2nd International Conference on Computer Science and Its Applications, Jeju, Korean, 10–12 December 2009; pp. 1–6.
20. Hermans, J.; Peeters, R. Private Yoking Proofs: Attacks, Models and New Provable Constructions. In Proceedings of the 8th International Conference on RFIDSec, Nijmegen, The Netherlands, 2–3 July 2012; pp. 96–108.
21. Lo, N.W.; Yeh, K.H. Anonymous coexistence proofs for RFID tags. *J. Inf. Sci. Eng.* **2010**, *26*, 1213–1230.
22. Ma, C.; Lin, J.; Wang, Y.; Shang, M. Offline RFID Grouping Proofs with Trusted Timestamps. In Proceedings of the International Conference on Trust, Security and Privacy in Computing and Communications, Liverpool, UK, 25–27 June 2012; pp. 674–681.
23. Bolotnyy, L.; Robins, G. Generalized "Yoking-Proofs" for a Group of RFID Tags. In Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services, San Jose, CA, USA, 17–21 July 2006; pp. 1–4.
24. Chien, H.Y.; Liu, S.B. Tree-Based RFID Yoking Proof. In Proceedings of the International Conference on Networks Security, Wireless Communications and Trusted Computing, Wuhan, China, 25–26 April 2009; pp. 550–553.
25. Ozcanhan, M.H.; Dalkilic, G.; Utku, S. Analysis of Two Protocols Using EPC Gen-2 Tags for Safe Inpatient Medication. In Proceedings of the Innovations in Intelligent Systems and Applications (INISTA), Albena, Bulgaria, 19–21 June 2013; pp. 1–6.
26. Peris-Lopez, P.; Orfila, A.; Hernandez-Castro, J.C.; Lubbe, J. Flaws on RFID grouping-proofs. Guidelines for future sound protocols. *J. Netw. Comput. Appl.* **2011**, *34*, 833–845.

27. Piramuthu, S. On Existence Proofs for Multiple RFID Tags. In Proceedings of the ACS/IEEE International Conference on Pervasive Services, Lyon, France, 26–29 June 2006; pp. 317–320.
28. Sundaresan, S.; Doss, R.; Piramuthu, S.; Zhou, W. A robust grouping proof protocol for RFID EPC C1G2 Tags. *J. Inf. Forensics Secur*. **2014**, *9*, 961–975.
29. Lien, Y.H.; Leng, X.; Mayes, K.; Chiu, J.H. Reading Order Independent Grouping Proof for RFID Tags. In Proceedings of the International Conference on Intelligence and Security Informatics, Taipei, Taiwan, 17–20 June 2008; pp. 128–136.
30. Jantarapatin, S.; Mitrpant, C.; Tantibundhit, C.; Nuamcherm, T.; Kovintavewat, P. Performance Comparison of the Authentication Protocols in RFID System. In Proceedings of the Management of Emergent Digital EcoSystems, New York, NY, USA, 26–26 October 2010; pp. 131–136.
31. Nuamcherm, T.; Kovintavewat, P.; Tantibundhit, C.; Ketprom, U.; Mitrpant, C. An Improved Proof for RFID Tags. In Proceedings of the International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, Krabi, Thailand, 14–17 May 2008; pp. 737–740.
32. Yen, Y.C.; Lo, N.W.; Wu, T.C. Two RFID-based solutions for secure inpatient medication administration. *J. Med. Syst.* **2012**, *36*, 2769–2778.
33. International Organization for Standardization. ISO/IEC 18000: Information Technology Automatic Identification and Data Capture Techniques—Radio Frequency Identification for Item Management Air Interface, 2003. Available online: http://www.iso.org/iso (accessed on 22 October 2015).
34. Cha, J.R.; Kim, J.H. Novel Anti-Collision Algorithms for Fast Object Identification in RFID System. In Proceedings of the 11th International Conference on Parallel and Distributed Systems, Fukuoka, Japan, 22–22 July 2005; pp. 63–67.
35. Zhai, J.; Wang, G. An Anti-Collision Algorithm Using Two-Functional Estimation for RFID Tags. In Proceedings of the International Conference on Computational Science and Its Applications, Singapore, 9–12 May 2005; pp. 702–711.
36. Fuentes, J.M.; Peris-Lopez, P.; Tapiador, J.E.; Pastrana, S. Probabilistic yoking proofs for large scale IoT systems. *Ad Hoc Netw*. **2015**, *32*, 43–52.
37. Leng, X.; Lien, Y.; Mayes, K.; Markantonakis, K. An RFID grouping proof protocol exploiting anti-collision algorithm for subgroup dividing. *Int. J. Secur. Netw.* **2010**, *5*, 79–86.
38. EPC™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 Mhz–960 Mhz, ver 2.0.0, EPCGlobal Inc., November 2013. Available online: http://www.gs1.org/sites/default/files/docs/epc/uhfc1g2_2_0_0_standard_20131101.pdf (accessed on 22 October 2015).
39. Bockorick, R.C.; Cooper, S.; Diorio, C.; Dressler, D.; Gutnik, V.; Hagen, C.; Hara, D.; Hass, T.; Humes, T.; Hyde, J.; *et al*. Design of ultra-low-cost UHF RFID tags for supply chain applications. *IEEE Comm. Mag*. **2004**, *42*, 140–151.
40. Xu, L.; Huang, C. Computation-efficient multicast key distribution. *IEEE Trans. Parallel Distrib. Syst*. **2008**, *19*, 577–587.
41. Balogh, J.; Békési, J.; Galambos, G. New lower bounds for certain classes of bin packing algorithms. *J. Theor. Comput. Sci.* **2012**, *440*, 1–13.
42. Fleszar, K.; Charalambous, C. Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem. *Eur. J. Oper. Res*. **2011**, *210*, 176–184.

43. Gupta, J.N.D.; Ho, J.C. A new heuristic algorithm for the one-dimensional bin-packing problem. *Prod. Plan. Control*. **1999**, *10*, 598–603.
44. Huang, H.H.; Ku, C.Y. A RFID grouping proof protocol for medication safety of inpatient. *J. Med. Syst*. **2009**, *33*, 467–474.
45. Peris-Lopez, P.; Orfila, A.; Mitrokotsa, A.; Lubbe, J. A Comprehensive RFID solution to enhance inpatient medication safety. *J. Med. Inf*. **2001**, *80*, 13–24.
46. Man, A.S.W.; Zhang, E.S.; Lau, V.K.N.; Tsui, C.Y.; Luong, H.C. Low Power VLSI Design for a RFID Passive Tag baseband System Enhanced with an AES Cryptography Engine. In Proceedings of the 1st Annual RFID Eurasia, Istanbul, Turkey, 5–6 September 2007; pp. 1–6.
47. Bogdanov, A.; Leander, G.; Paar, C.; Poschmann, A.; RobshawM.J.B.; Seurin, Y. *Hash Functions and RFID Tags: Mind the Gap. Cryptographic Hardware and Embedded Systems*; Springer Berlin Heidelberg: Heidelberg, Germany, 2008; pp. 283–299.
48. Feldhofer, M.; Wolkerstorfer, J. Strong Crypto for RFID Tags—A Comparison of Low-Power Hardware Implementations. In Proceedings of the IEEE International Symposium on Circuits and Systems, New Orleans, LA, USA, 27–30 May 2007; pp. 1839–1842.
49. Bo, S.; Ito, Y.; Nakano, K. CRT-Based DSP Decryption Using Montgomery Modular Multiplication on the FPGA. In Proceedings of the IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, Shanghai, China, 16–20 May 2002; pp. 532–541.
50. Tsoi, K.H.; Leung, K.H.; Leong, P.H. W. Compact FPGA-based true and pseudo random number generators. In Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, CA, USA, 9–11 April 2003; pp. 51–61.
51. Nakano, K.; Kawakami, K.; Shigemoto, K. RSA Encryption and Decryption Using the Redundant Number System on the FPGA. In Proceedings of the IEEE International Symposium on Parallel & Distributed Processing, Rome, Italia, 23–29 May 2009; pp. 1–8.