Secure and Practical Key Distribution for RFID-Enabled Supply Chains*

Tieyan Li[1], Yingjiu Li[2], and Guilin Wang[3]

[1] Irdeto (Cloakware) Beijing, China,
li.tieyan@irdeto.com
[2] School of Information Systems, Singapore Management University, Singapore 178902,
yjli@smu.edu.sg
[3] Centre for Computer and Information Security Research, School of Computer Science and Software Engineering, University of Wollongong, Wollongong, NSW 2522, Australia,
Guilin@uow.edu.au

Abstract. In this paper, we present a fine-grained view of an RFID-enabled supply chain and tackle the secure key distribution problem on a *peer-to-peer* base. In our model, we focus on any pair of consecutive parties along a supply chain, who agreed on a transaction and based on which, certain RFID-tagged goods are to be transferred by a third party from one party to the other as in common supply chain practice. Under a strong adversary model, we identify and define the security requirements with those parties during the delivery process. To meet the security goal, we first propose a resilient secret sharing (RSS) scheme for key distribution among the three parties and formally prove its security against *privacy* and *robustness* adversaries. In our construction, the shared (and recovered) secrets can further be utilized properly on providing other desirable security properties such as tag authenticity, accessibility and privacy protection. Compared with existing approaches, our work is more resilient, secure and provides richer features in supply chain practice. Moreover, we discuss the parameterization issues and show the flexibility on applying our work in real-world deployments.

Key words: RFID, security, privacy, key distribution, secret sharing

1 Introduction

Radio-frequency identification (RFID) is a wireless Automatic Identification and Data Capture (AIDC) technology that has been widely deployed in many applications, especially in supply chain management. For dynamic RFID-enabled supply chains, the parties in a supply chain are usually lack of pre-existing trusted relationships. Unfortunately, almost all existing RFID *privacy-enhanced*

* This version is slightly different from the one published in Securecomm 2011. In particular, clearer comments on unlinkability are added.

authentication protocols such as [9, 8, 6], assuming a central database on managing all secret keys of the tags, may fail on delivering key information to the correct parties when a large scale of RFID tags move along dynamic supply chains.

A practical solution to the key distribution problem is the secret sharing approach, where a tag key is split into a number of shares and the shares are stored in multiple tags. Since the tag keys are stored in the tags directly, an authorized party can collect enough shares and recover the keys, while an adversary is assumed to have limited access to the tags such that s/he cannot collect enough shares for recovering the keys. Since there is no need of distributing key information among supply chain parties, this approach is particularly useful for protecting RFID tags in dynamic supply chains.

A recent work in this direction is conducted by Juels, Pappu and Parno [4], which we call the JPP mechanism for short. In this solution, a common key for a batch of tags is split with (k, n)-Tiny Secret Sharing (TSS) scheme, and each tag stores a tiny share together with its individual (encrypted) information. A reader can recover the common key with access to at least k shares, and then decrypt the information on each tag in the batch. Since the share is tiny enough to fit in EPC tag, the proposed scheme is claimed to be suitable for practical RFID-enabled supply chains. However, the JPP mechanism poses a threat on the tags due to its *weak* adversary model, of which anyone who can scan the tags, can recover the secret. Therefore, an adversary has the intension to stay close to the tags for the convenience of scanning them, and recover the secret for easy cloning the whole batch of the tags.

On observing the hardness on designing and deploying a uniform security solution to multiple supply chain parties across geographically distinct organizations, we tackle the security problem with goods delivery in RFID-enabled supply chains from a focalized viewpoint. We look into the minimal (usually transaction-based) unit of any supply chain on processing RFID-tagged goods and focus on the security needs arisen from the involved parties. Based on the unique view, we make three major contributions in this paper.

1. We focus on any pair of consecutive parties linked by a transaction and a third party who delivers goods from one party to the other (as in common supply chain practice, usually referred to as third party logistics, or 3PL). We then identify and define the security requirements among those parties during goods delivery under a strong adversary model.
2. We propose a resilient secret sharing (RSS) scheme for key distribution among the three parties and prove its security in a formal way against both *privacy* and *robustness* adversaries.
3. We design a specific construction using the RSS scheme, so that the shared secrets are further utilized on providing additional security properties (beyond key distribution) such as tag authentication, accessibility and privacy protection.

Compared with relevant approaches, our work demonstrates a number of advantages in terms of resiliency, security and flexibility. Also, the generic scheme and the specific construction proposed in this paper, are easily extendible for their deployments in any realistic supply chain scenario.

The rest of this paper is organized as follows. In Section 2, we review secret sharing approaches in RFID security realm. In Section 3, we describe a scenario on secure goods delivery and the security properties associated with it. We elaborate on our resilient secret sharing (RSS) scheme and prove its security in Section 4. Following on, we present our construction based on the RSS scheme in Section 5, and discuss parameterization issues in Section 6. Finally, we conclude this paper and point out the future works.

2 Secret Sharing Approaches

On solving the key distribution problem in RFID-enabled supply chains, two major secret sharing based approaches [5, 4] were proposed.

The first work is the "Shamir Tag" [5] proposed by Langheinrich and Marti. Based on a weak (*w.r.t.,* "hit-and-run") adversary model, the authors devised the secret sharing mechanism to distribute the *true* ID of a tag across time and space separately. The time-based mechanism splits the ID of a tag with Shamir's secret sharing scheme [12], and stores all the shares on the tag itself. Being queried, the shares are released gradually and once all bit values of the shares are collected, can the original ID of a tag be computed. However, the practicability on applying the proposed mechanisms in supply chain is questionable as it takes too long on identifying a tag.

In USENIX Security 08, Juels, Pappu and Parno proposed a key sharing mechanism (*w.r.t.,* the JPP mechanism) [4] to enhance the practicality of the early solution [5] by removing the constraints on the period of each tag being read and the number of tags attached to each item. In the JPP mechanism, a batch of tags share the same secret key, which is split into n shares using a (k, n) tiny secret sharing (TSS) scheme, where $k < n$ is a threshold. Anyone who collects at least k shares can recover the secret. Similarly, the JPP mechanism provides two solutions for its (k, n)-TSS scheme: one is "secret sharing across space", the other is "secret sharing across time".

The JPP mechanism is particularly efficient for ownership transfer in RFID-enabled supply chains since it eliminates the need for distributing a database of tag keys among supply chain parties. The scheme is secure under the assumption that an adversary cannot get access to enough shares for recovering a tag key in the "open area" (*e.g.,* retail stores or customer homes), while legitimate supply chain parties can collect enough shares for recovering each tag key in the "closed area" of a supply chain, to which the adversary does not have access. However, the JPP mechanism works in *end-to-end* principle (regarding the starting and ending points of a tagged item moving through a supply chain) that makes intermediate supply chain parties unable to adjust the threshold of shares collected in recovering tag keys. This renders the proposal either impractical *w.r.t.,* stronger adversary or insecure due to tag cloning attack.

We realized that it could be hard to deal with the complex security needs of multiple parties (normally across multiple geographical and political regions) in a global supply chain, we thus target on two adjacent parties in any supply chain and a third party who transfers goods for these two parties. In common practice in supply chain management, assuming these parties know each other via some trust relationships such as a signed contract for their business transactions, is more reasonable. We stress that such a *peer-to-peer* view of a supply chain facilitates a more realistic and practical model than the global view, and based on what, more precise security requirements can be defined and fulfilled. We depict below such a scenario on delivering goods assisted with RFID technology.

3 Security Properties

In this section, we take for example a typical case for batch goods delivery as used in standard supply chain practice. We then define the security properties for each of the roles based on such a scenario.

3.1 Batch Goods Delivery Scenario

We consider three different roles in a simplified model: Alice, Bob and Carol. Alice, denoted by **A**, is the sender of a batch of goods (*e.g.* a manufacturer); Bob, denoted by **B**, is the receiver of the batch (*e.g.* a distributor who receives the goods from **A**); and Carol, denoted by **C**, is the Third Party Logistics (3PL) partner (*e.g.,* a transporter or carrier of the goods from **A** to **B**).

Suppose **A** and **B** (and **C**) signed contracts for the purchase and delivery of some goods beforehand. Now, the goods must be delivered securely from **A** to **B** by **C** to fulfill the contracts. If each item of the goods is attached with an RFID tag, a supply chain party can process the goods in an efficient way (by scanning all items once in a whole). It is also desirable to provide necessary security features such as anti-cloning without incurring much additional cost.

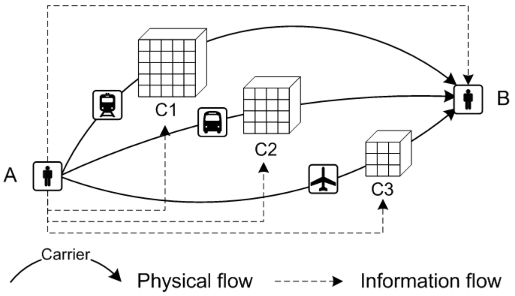

Fig. 1. Batch Goods Delivery from **A** to **B**, by **C**.

As an example, Fig. 1 illustrates the scenario where a batch of 50 tags are packed into 3 cases, including a 5 × 5 case, a 4 × 4 case and a 3 × 3 case. These cases are delivered from **A** to **B** via different physical flows by **C** (including C1, C2, C3, respectively). Some shared information prepared by **A** can be sent to **B** and **C** separately via the information flow. Facilitated with RFID technology, **C** can scan all RFID tags periodically during delivery, until all of them arrive at **B**. The scanning can be used to check the existence of all tags in the batch. If any adversary exists in the delivery path, however, s/he may clone some tags and thus replace authentic goods with counterfeited goods even though the adversary may not be able to know the secret keys for decrypting the tags' contents as in the JPP mechanism. To address this concern, we propose to tackle the RFID tags authentication problem in which nobody (even **C**) except **A** and **B** can access the content of tags while **C** is enabled to check the existence of all or most of the tags in the batch conveniently.

We identify two kinds of flows during the process of goods delivery. One is the physical flow in which the goods are transported by **C** through containers on ships or trucks. The other is the information flow between **A**, **B** and **C**. Intuitively, we utilize the information collected from both flows for the purpose of achieving desired security properties.

3.2 Desired Security Properties

We further identify the following security properties for guaranteeing secure goods delivery in above scenario.

- ◇ **Authenticity of tags in cases. C** wants to authenticate the tags case by case periodically. Other than authentication purpose, **C** has no more advantage for her to access or even clone the tags. Both group authentication (case based), and individual tag authentication are demanded for efficiency or accuracy.
- ◇ **Authenticity of tags in batch. B** wants to authenticate the batch of tags in a whole as the final verification. Being the new owner of the tags, **B** shall grasp all (secret) information about the tags which include the ability to update the tags.
- ◇ **Accessibility of individual tags.** Only **B** can obtain the secret information for the accessibility of individual tags. **C** or other adversaries cannot access or clone those tags.
- ◇ **Privacy protection** In the sense of protecting the tags' identifiers, all tags' IDs are encrypted by a secret key, which can only be recovered by **B**. Without necessary authorization, **C** can not even access the encrypted information, let along the decryption of such information. Though, by scanning the tags, **C** or any reader can still track the tags by the unique or unchanged temporary identities within the current peers, it cannot further track the tags before or after the current distribution.

In fact, the fundamental building block for all above listed properties is the "key distribution" problem among **A**, **B** and **C**. By designing a resilient secret sharing (RSS) scheme in next section, we can make a nice construction for above scenario so that **A** and **C** share a secret key solely for the authentication purpose, and **A** and **B** share another key for all other security properties.

4 Resilient Secret Sharing Scheme

In order to achieve all the security properties, we work on key distribution first and design a resilient secret sharing (RSS) scheme inspired by the JPP mechanism. The proposed scheme contains the merits of the JPP mechanism in terms of tiny shares and error-correcting code based secret sharing algorithm. It also enhances the JPP mechanism with resiliency by collecting the shares from two different sources (*w.r.t.,* physical flow and information flow).

4.1 Preliminaries

We recall the JPP mechanism [4]: an n-party secret-sharing scheme is a pair of algorithms $\mathbf{\Pi} = (\mathsf{Share}, \mathsf{Recover})$ that operates over a message space $\mathbb{X}$, where

- ◇ Share is a deterministic algorithm (if a fixed Reed-Solomon code is used, but a probabilistic algorithm in [12]) that takes input $x \in \mathbb{X}$ and output the n-vector $S \leftarrow_R \mathsf{Share}(x)$, where $S_i \in \{0,1\}^*$. On invalid input $\hat{x} \notin \mathbb{X}$, Share outputs an n-vector of the special ("undefined") symbol $\perp$.
- ◇ $\mathsf{Recover}$ is a deterministic algorithm that takes input $S \in (\{0,1\}^* \bigcup \diamondsuit)^n$, where $\diamondsuit$ represents a share that has been erased (or is otherwise unavailable). The output $\mathsf{Recover}(S) \in \mathbb{X} \bigcup \perp$, where $\perp$ is a distinguished value indicating a recovery failure.

Utilizing Error Correcting Code (ECC), a generalization of the secret sharing schemes is defined as $\mathbf{\Pi}^{ECC} = (\mathsf{Share}^{ECC}, \mathsf{Recover}^{ECC})$. An $(n,k,d)_Q$-ECC operates over an alphabet Σ of size $|\Sigma| = Q$. Share^{ECC} maps $\Sigma^k \to \Sigma^n$ such that the minimum Hamming distance in symbols between (valid) output vectors is d. For such a share function (Share^{ECC}), there is a corresponding recover function ($\mathsf{Recover}^{ECC}$) that recovers a message successfully with up to $(d-1)/2$ errors or $d-1$ erasures.

In our adversary model[2], we consider two security requirements of secret sharing: *privacy* and *robustness*. Given a limited number of shares, an attack against privacy aims to recover the secret x shared among n parties. A robustness attacker tries to tamper a number of shares such that a legal user cannot recover the correct secret x. In formal, we give the following definitions.

Privacy An ordinary adversary can actively attack the communication links. In our scenario of secure goods delivery, as did in [4] we focus on $\mathsf{underinformed}$ $\mathsf{adversary}$, who has access to limited number of shares. Informally, privacy require that such an underinformed adversary should not be able to recover the secret

[2] Our adversary model is adapted from [4], which in turn is obtained by extending the model given in [1].

unless he can get access to at least k correct shares. However, since we are working on *gradated*, rather than *perfect* or *computational* secret sharing schemes, an adversary with limited number of shares may be able to get *partial* information about the secret, though it cannot completely recover the secret itself. Moreover, the more the shares the adversary gets, the more information it reveals. In the following formal definition of privacy, oracle $\mathsf{corrupt}(S, i)$ is defined as a function of (S, i), *i.e.,* when the adversary submits i it will get S_i, the i-th share of a secret x.

Definition 1 (Privacy) *Formally, we say a (k, n)-RSS scheme* ($\mathbf{\Pi}, \mathbb{X}$) *satisfies* $(q_p, t_p, \varepsilon_p)$-$\mathsf{privacy\ w.r.t.\ underinformed\ attackers}$, *if for any adversary* $\mathcal{A}$ *who can make* q_p *corrupt queries to acquire* q_p *shares (*S_p *denotes the set of these* q_p *shares) corresponding to a shared secret* x *and can run within the time of* t_p, $\mathcal{A}$*'s advantage to win the following experiment* $\boldsymbol{Exp}_{\mathcal{A}}^{Pri}$, *i.e.,* $\boldsymbol{Exp}_{\mathcal{A}}^{Pri}$ *outputs* 1, *is not greater than* ε_p*:*

$$\mathrm{Adv}_{\mathcal{A}}^{Pri}[\mathbf{\Pi}, \mathbb{X}] \overset{\triangle}{=} \Pr[\mathbf{Exp}_{\mathcal{A}}^{Pri} = 1] \leq \varepsilon_p. \tag{1}$$

```
Experiment $\mathbf{Exp}_{\mathcal{A}}^{Pri}$
  1) $x \leftarrow_R \mathbb{X}$
  2) $S = (S_1, \cdots, S_i, \cdots, S_n) \leftarrow \mathsf{Share}(x)$
  3) $x' \leftarrow \mathcal{A}^{\mathsf{corrupt}(S,\cdot)}(S_p : S_p \subset S \wedge |S_p| = q_p)$
  4) Return '1' if $x = x'$, else '0'.
```

Privacy Experiment

Note that different from computational secret sharing schemes, here (as well as in Definition 2) we don't specify the adversary $\mathcal{A}$ should be a probabilistic polynomial time (PPT) algorithm. In contrast to the privacy definition given in [4], we add the running time r_p to parameterize an adversary. This makes our definition more flexible and general, though our concrete scheme is secure regardless the adversary's running time (refer to Theorem 1). Moreover, we notice that the indistinguishability game specified in Appendix B.1 of [4] is too strong to be satisfied by ECC-based secret sharing schemes. The reason is that given two secrets κ^0 and κ^1, adversary $\mathcal{A}$ can first run the encoding algorithm to regenerate the corresponding codewords S^0 and S^1, i.e. the shares for κ^0 and κ^1 respectively. As S^0 and S^1 must differ from each other for at least one index, say j, then $\mathcal{A}$ makes oracle query $\mathsf{corrupt}(S^b, j)$ to get S_j^b. Finally, to win the game $\mathcal{A}$ only needs to trivially guess $b = 1$ iff $S_j^b \in S^1$.

Robustness Informally, robustness means that the original secret can be recovered even if the adversary has tampered some of the shares corresponding to such a shared secret.

Definition 2 (Robustness) *Formally, we say a (k,n) RSS scheme* ($\mathbf{\Pi}, \mathbb{X}$) *is* $(q_r, t_r, \varepsilon_r)$-$\mathsf{robust}$, *if for any adversary* $\mathcal{A}$ *who can make* q_r *corrupt queries to* *get and tamper q_r shares (those original and tampered q_r shares form sets S'_r and S''_r respectively) of a shared secret x, which is selected by $\mathcal{A}$ itself, and has running time within t_r, $\mathcal{A}$'s advantage to win the following experiment $\boldsymbol{Exp}_{\mathcal{A}}^{Rob}$, i.e., $\boldsymbol{Exp}_{\mathcal{A}}^{Rob}$ outputs 1, is not greater than ε_r:*

$$\mathrm{Adv}_{\mathcal{A}}^{Rob}[\mathbf{\Pi}, \mathbb{X}] \overset{\triangle}{=} \Pr[\mathbf{Exp}_{\mathcal{A}}^{Rob} = 1] \leq \varepsilon_r. \tag{2}$$

```
Experiment $\mathbf{Exp}_{\mathcal{A}}^{Rob}$
  1) $x \leftarrow \mathcal{A}$, where $x \in \mathbb{X}$
  2) $S = (S_1, \cdots, S_i, \cdots, S_n) \leftarrow \mathsf{Share}(x)$
  3) $S''_r \leftarrow \mathcal{A}^{\mathsf{corrupt}(S,\cdot)}$, where $|S''_r| = q_r$
  4) $x' \leftarrow \mathsf{Recover}\{S''_r \cup (S - S'_r)\}$
  5) Return '1' if $x \neq x'$, else '0'.
```

Robustness Experiment

4.2 RSS

Let's first review McEliece's secret sharing scheme based on Reed-Solomon (RS) codes [7]. Let $B = (b_1, b_2, \ldots, b_k)$ be the secret, where b_i is an m-bit symbol in $\mathbf{GF}(2^m)$. There exists a unique codeword D in the (k, n)-RS code $(n < 2^m)$ with $D = (d_1, d_2, \ldots, d_n)$, where $d_i = b_i$ for $1 \leq i \leq k$. Only the rest $n - k$ symbols $\{d_i | (k + 1 \leq i \leq n)\}$ are available for distribution to those sharing the secret. Of all shares, at least k shares are required to recover the secret.

On a high level, our RSS scheme aims at achieving resiliency by combining shares from both physical flow and information flow. Suppose we have only one case containing r tags in the physical flow, and a database[3] as the source of an information flow. We naturally assign one portion of the shares (typically one share for each tag) on the tags, and keep the other portion of the shares in the database. To this end, for a (k, n)-RS code, we can assign r shares to r tags and $n - k - r$ shares to the database (assuming $r < n - k$). Further on, we require that any single flow can not contribute enough shares on recovering the secret (so, $r < k$ and $n - k - r < k$). Thus, we roughly ensure the resiliency of the RSS scheme on recovering a secret with shares contributed from both flows. Such an RSS scheme can be illustrated in Fig. 2 as below.

Ideally, all r tags in a case can be scanned for sorting out all r shares. However, 100% reading is not typically guaranteed in practice as there always be some (*e.g.,* $2 - 3\%$) reading failures in realistic RFID deployments. Suppose all but δ tags are correctly scanned, we can obtain up to $r - \delta$ shares from the readings. For tolerating the reading errors, our RSS scheme allows more shares

[3] Note that an online database is not required in our scenario, as a partner's database (*e.g.*, Partner **A** in Fig. 1) is only used to store the shares and pass the shares down (to **B** and **C**) all in once.

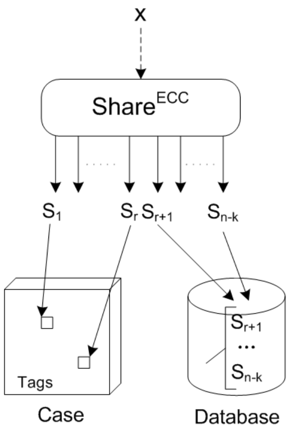

Fig. 2. RSS scheme. A secret x is shared into $n-k$ available shares, in which r shares are distributed into r tags respectively and the other $n-k-r$ shares are stored in a database.

contributed from the information flow to compensate the missing shares in the physical flow. To ensure our RSS scheme having this resiliency, δ more shares are required to be stored in the database.

In ECC based secret sharing scheme, a share is a symbol in a codeword (*e.g.,* in RS code), which is much shorter than the original secret. An adversary could launch a guessing attack on trying all the possibilities of a missing share. For instance, for a RS-code on $\mathbf{GF}(2^m)$, such a guessing attack needs 2^m brute force trials. To defend against the guessing attack from attackers who are able to scan all tags, the RSS scheme requires at least t shares contributed by the server in any recovery operation. Thus, a brute force guessing attack may take at $2^{t\times m}$ to recover the secret. I.e., if the system security parameter is set at 128 bits long, and $m = 16$, then we have $t = 8$.

Combining above two requirements, the RSS scheme allocates at a minimum t shares and a maximum $t+\delta$ shares to be stored in the database. Since we don't want the server to calculate the secret along by its shares, or even for brute force attack, we require that $k \geq 2t+\delta$. Our assumption is that the server may either collect all tags in a case and be able to scan $r \sim r-\delta$ tags, or collecting no tag/case at all. As we require that the combination of $r+t$ shares from the tags and database is enough to recover the original secret, so we set the threshold $k = r+t$. Also, $k \geq 2t+\delta$ implies that the number of tags $r \geq t+\delta$. Otherwise, the server is able to launch guessing attacks for guessing up to $t-1$ shares.

Definition 3. *A $(k,n)_{m,t,r,\delta}$-RSS scheme is a tuple $(\mathbf{\Pi^{ECC}}, \mathbb{X})$, satisfying $t \times m \geq \tau$ (τ is the security parameter of the system), $k = r+t$ and $r \geq t+\delta$, $n = 2r+2t+\delta$; such that $\mathbf{\Pi^{ECC}}$ distributes $n-k$ shares of a secret $x \in \mathbb{X}$, to the tags (totally r shares) and to the database (totally $t+\delta$ shares). Collecting $r-\delta \sim r$ shares from tags, and correspondingly $t+\delta \sim t$ shares from the database, suffices to recover x.*

On the security of the defined RSS scheme above, we have the following

Theorem 1. (a) *For the* $(k,n)_{m,t,r,\delta}$*-RSS scheme* $(\mathbf{\Pi^{ECC}},\mathbb{X})$*, any underinformed adversary* $\mathcal{A}$*'s advantage is bounded by* ε_p *such that*

$$\mathrm{Adv}_{\mathcal{A}}^{Pri}[\mathbf{\Pi}^{ECC},\mathbb{X}] \leq \varepsilon_p \leq 1/2^{m(k-q_p)}, \qquad (3)$$

where $q_p \leq k = r + t$*.*
(b) *For the* $(k,n)_{m,t,r,\delta}$*-RSS scheme* $(\mathbf{\Pi^{ECC}},\mathbb{X})$*, any adversary* $\mathcal{A}$ *with unbounded running time and making up to* $q_r \leq d/2$ *(or* $q_r \leq \lfloor (d-1)/2 \rfloor$*) corruptions has advantage zero to win the experiment* $\boldsymbol{Exp}_{\mathcal{A}}^{Rob}$*. Namely,*

$$\mathrm{Adv}_{\mathcal{A}}^{Rob}[\mathbf{\Pi}^{ECC},\mathbb{X}] = \varepsilon_r = 0. \qquad (4)$$

Proof (a) For any underinformed adversary $\mathcal{A}$ who has made $q_p \leq k = r + t$ corruptions, its total amount of information about the original secret x is upperbounded by $(2^m)^{q_p}$. More specifically, for Reed-Solomon code, the adversary $\mathcal{A}$ can only get q_p linear equations to solve the k unknown elements in field $\mathbf{GF}(2^m)$ (*i.e.,* k components of x [11]). As $q_p \leq k$, regardless $\mathcal{A}$'s running time its advantage ε_p to derive the secret x is bounded by $1/2^{m(k-q_p)}$. So, Eq. (3) follows.

(b) The result on robustness comes from the nature of Reed-Solomon code, as it is an error-correcting code. Namely, an error-correcting code with design distance d can be used to correct up to $d/2$ errors. Here, the adversary $\mathcal{A}$ has tampered $q_r \leq d/2$ symbols. So, using any popular decoding algorithm (*e.g.,* the decoding algorithm for alternant codes, specified on page 403 of [11]), these errors can be identified and corrected efficiently. In other words, the legal user (e.g. party **B** in our secure goods delivery scenario) is always able to recover the original secret x from shares mixed with those tampered ones. Therefore, regardless $\mathcal{A}$'s running time its advantage ε_r in the experiment $\mathbf{Exp}_{\mathcal{A}}^{Rob}$ is zero. That is, Eq. (4) holds for any $q_r \leq d/2$ (or $q_r \leq \lfloor (d-1)/2 \rfloor$). ■

5 Our Construction

Above we give a generalized definition and security proof of the RSS scheme, in what follows we elaborate the constructions on applying the RSS scheme in a typical case of secure goods delivery with batch RFID tags.

In our simplified example, we suppose there are totally R tags attached on goods as a batch to be transferred from **A** to **B**, via **C**. The tags in the batch are allocated equally into l cases, each having r tags ($R = l \times r$). We assume the batch has a suitable size such that r or R is not too big to be contained, otherwise we can consider a batch as a number of blocks with suitable sizes, which are to be processed as one unit. We then discuss a tag belonging to both a case and a batch. More details are discussed in Section 6.

5.1 Secret Generation and Sharing

Before the delivery of goods, **A** generates the secrets x for the specific case and y for the whole batch such that $x, y \in \mathbb{X}$ and $|x| = |y| = \tau$, where τ is the security parameter of the system.

At the case level, **A** employs a $(k, n)_{m,t,r,\delta}$-RSS scheme according to the definition introduced in Section 4, to distribute the case secret x. For all r tags in a case, **A** assigns one share to each tag. **A** also assigns $t + \delta$ shares to **C** to facilitate the verification by **C** on such a case during delivery.

Similarly, at the batch level, **A** employs a $(K, N)_{m,t,R,\varDelta}$-RSS scheme to distribute the batch secret y, assuming the security parameter and the size of the shares are not changed. For all R tags in a batch, **A** assigns one share to each tag. **A** then assigns $t + \varDelta$ shares to **B** to facilitate the verification by **B** on the whole batch.

With this setting, a tag is assigned two shares: one for the case and one for the batch. Collecting the shares from the tags, **C** or **B** can recover the case secret or batch secret respectively, together with their contributed shares given by **A**. The schematic of the RSS construction is illustrated in Fig. 3.

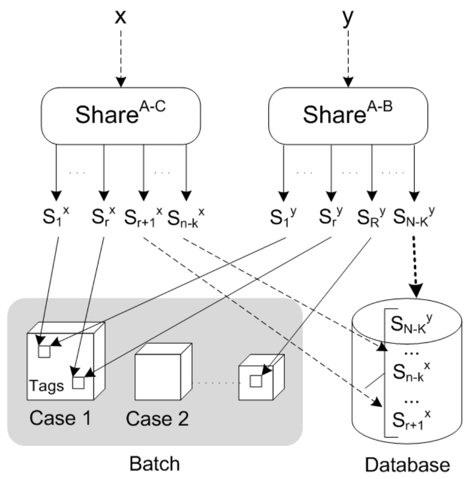

Fig. 3. Schematic of RSS construction. The case secret x is shared for Case 1, in which r shares are distributed into tags and the rest $n-k-r$ shares are stored in the database (to be assigned to **C**); the batch secret y is shared, in which R shares are distributed into all tags in the batch and the rest $N - K - R$ shares are stored in the database (to be assigned to **B**). Thus, an encoded tag carries 2 shares.

5.2 Tag Encoding

For a specific tag i, we obtain its case share $S_i^x \leftarrow \mathsf{Share}^{\mathbf{A-C}}(x)$ $(1 \leq i \leq r)$ and batch share $S_j^y \leftarrow \mathsf{Share}^{\mathbf{A-B}}(y)$ $(1 \leq j \leq R)$. As we work on $\mathbf{GF}(2^m)$, the size of a share could be $m = 16$ bits which is tiny (*e.g.,* 32 bits in total for carrying 2 shares [4]) and suitable to be embedded into an EPC C1G2 tag. In practice, we shall prepare another 16 bits (or less) for making the shares in an ordered sequence. Thus, for an EPC C1G2 tag, we can assign 48 lower significant bits (LSBs) of the EPC memory bank for storing the sequence number and the shares. For the other 48 bits, we can either leave them untouched for classification purpose, or fill them with arbitrary random value for privacy purpose. Now we denote the current value in the EPC memory as the pseudo-ID (or PID) of a tag.

Tag ID Encryption Suppose the original 96-bit EPC code, denoted as ID, is moved from the "EPC Memory" Bank to the "User Memory" Bank. To provide privacy protection to the EPC code, we store it in encrypted form, so that no one can decrypt and obtain the original code without a proper key. As **B** will be the next owner of the tags, we assign **B** the appropriate role of possessing the proper secret to decrypt the real IDs of those tags. Since y is the only secret shared and known between **A** and **B**, we derive the encryption key e from y such that $e = H(y)$, where $H(.)$ is a cryptographic secure hash function. Then we use e to encrypt the EPC code (in any authenticated encryption mode) and obtain the encrypted and authenticated message $\widetilde{ID} = Enc\_Auth(e, ID)$, where $Enc\_Auth(.)$ is the authenticated encryption algorithm.

Tag PIN Generation To achieve the authentication purpose, a tag's Access and Kill PINs, denoted as $APIN$ and $KPIN$, are serving as the authenticators by **C** or **B** on performing PIN-based authentication protocol as in [3]. Slightly different from the protocol [3] on using a full (32-bit) Access PIN or Kill PIN for authentication purpose, we hereby use the two halves of Access and Kill PINs of a tag for the same purpose, such that **C** is refrained from either access or kill a tag with its knowledge on the halves of PINs. While **B** can still authenticate and access a tag individually, by deriving the full Access and Kill PINs. Apparently, we can derive the PINs using the secrets (x and y) shared between **A**, **B** and **C**. Note that various constructions are possible, we only introduce a specific construction achieving above security properties for **B** and **C**. On a high level, we generate half APIN and half KPIN with **C**'s secret x, and the other halves with **B**'s secret y. We compute $\kappa_{\mathbf{C}} = H(x||PID)$ and $\kappa_{\mathbf{B}} = H(y||PID)$ for a tag by reading its PID. We assign 16 lowest significant bits (LSBs) of $\kappa_{\mathbf{C}}$ as the 16 LSBs of $APIN$ and the other 16 most significant bits (MSBs) of $\kappa_{\mathbf{C}}$ as the 16 LSBs of $KPIN$. Also, we assign 16 LSBs of $\kappa_{\mathbf{B}}$ as the 16 MSBs of $APIN$ and the other 16 MSBs of $\kappa_{\mathbf{B}}$ as the 16 MSBs of $KPIN$. Thus, we have

$$APIN = [APIN]_{31:16}||[APIN]_{15:0} = [\kappa_{\mathbf{B}}]_{15:0}||[\kappa_{\mathbf{C}}]_{15:0}$$
$$KPIN = [KPIN]_{31:16}||[KPIN]_{15:0} = [\kappa_{\mathbf{B}}]_{31:16}||[\kappa_{\mathbf{C}}]_{31:16}$$

Note that for **C** to conduct the PIN-based authentication, we expect a positive or negative result from the tag indicating whether the correct halves of Access and Kill PINs are presented to it[4].

We are now ready to encode the tag by writing all generated codes into a tag. Again, we write on tag the shares in the EPC memory, the encrypted EPC code in the user memory, and the access and kill PINs in the reserved memory.

5.3 Secret Recovery and Verification

During delivery, **C** would verify the tags in a case from time to time. Suppose the total number of collected shares in a case is p, if $r - \delta \leq p \leq r$, **C** can recover the secret x by contributing up to $t + r - p$ shares; if not, there is no enough shares for **C** to recover the secret. Based on the secret value, **C** can generate the halves of Access and Kill PINs for each tag as described above. **C** can then authenticate each tag by performing the half-PIN-based authentication protocol described above.

When all goods are delivered to **B**, **B** would verify the tags in the batch. Similarly, suppose **B** collects P shares from all the cases. if $R - \Delta \leq P \leq R$, **B** can recover the secret y by contributing up to $t + R - P$ shares; if not, there is no enough shares for **B** to recover the secret. Based on the secret value, **B** can generate the other halves of Access and Kill PINs for each tag in the batch. **B** can obtain from **C** the halves of Access and Kill PINs of each tag based on a case, or generate by itself the half PINs by collecting all shares from **C**.

Whatsoever, **B** can access all the tags and even kill all the tags as the new owner. Suppose **B** accesses a tag and reads its encrypted ID ($\widetilde{ID}$), **B** can decrypt and authenticate it with e from y and obtain the original EPC code of the tag.

5.4 Analysis and Comparison

We summarize the desired security properties in secure goods delivery and show how they are achieved in our construction using the RSS scheme.

- ▶ **Key distribution.** Our RSS scheme ensures that only **B** and **C** can derive the secrets they shared with **A**. Without additional share(s) from **B** and **C**, no adversary can derive any secret by solely collecting shares from tags. **A** securely distributes the secrets to **B** and **C** via both physical and information flows.
- ▶ **Authenticity.** **C** can verify that most tags in a batch or case are presented, and individually, every tag can be authenticated by **C** via half-PIN-based authentication. Similarly, **B** can verify the whole batch together and authenticate individual tags one by one.

[4] Although not fully conforming with current EPC C1 G2 specification, we argue that achieving above half-PIN-based authentication on a tag is rather simple with a re-designed circuit on the PIN logic, which is practical and costless.

- ► **Accessibility & Anti-cloning.** Only **B** can derive full Access and Kill PINs for all of the tags in a batch, and thus can access the tags with proper PINs. No adversary, including **C**, can derive the secrets and full PINs for accessing or cloning the tags.
- ► **Privacy Protection.** Only **B** can obtain the original EPC code of a tag. The privacy of the tag identifier is protected against **C** or any adversary. The privacy problem of tracking the pseudo-ID of a tag is addressed for key distributions between different peers (not within the same peers) since different PID's of a tag are supposed to be used in such cases.

The secret sharing approaches present a new research direction on solving the key distribution problem in RFID-enabled supply chains. Although the JPP mechanism is the first applicable solution for RFID-enabled supply chains without pre-sharing of secrets, it's security level is not sufficient as mentioned earlier. Our RSS scheme improves the security with additional shares contributed from the information flow. Other than key distribution, our RSS construction provides more desired security properties such as anti-cloning than the JPP mechanism.

The advantages of secret sharing approaches can be clearly demonstrated by comparing with existing RFID authentication protocols [9, 8, 6]. These authentication protocols are designed to have different security and efficiency features with a common assumption that shared keys must exist between mutually trusted parties, and that the tag keys are stored in a central database. Another difference is that the protocol messages in these protocols are unlinkable between authentication sessions.

Basically, all of them achieves authentication on individual tags, but not on a batch of tags. Moreover, in a strong adversary model where tags can be corrupted, all except our RSS scheme fail on providing anti-cloning feature as the tags' secrets are disclosed. Table 1 lists the major security features of our scheme in comparison with traditional schemes.

	Key Storage (DB/Tag)	Authentication (Group/Tag)	Anti-Cloning (Tag Corruption)	Type of Privacy (Unlinkability/ID Secrecy)
[9][8][6]	Central DB	Tag	No	Unlinkability betw. auth. sessions
TSS [4]	Tag	Group	No	ID Secrecy
RSS	Partner DB & Tag	Group & Tag	Yes	Unlinkability betw. diff. peers

Table 1. Comparison of Security Properties

6 Parameterization

In real-world implementation, the "Philips UCODE" Gen2 tag can be employed. The tag has 512 bits of on-chip memory, containing a 96-bit EPC memory, a

32-bit TID memory, a 128-bit programmable user memory and a 64-bit reserved memory for storing Access and Kill PINs. As required by our scheme, we shall replace the original EPC code with the shares in EPC memory, and store the encrypted (and authenticated) EPC code into the user memory.

As a running example, we suppose there are totally 100 tags in a batch which are packed equally into 5 cases each having 20 tags exactly. At the case level, our RSS scheme employs a $(28, 60)$-RSS Scheme so that given a case, we need to collect at least 28 shares to recover the case secret. Our scheme works over the field $GF(2^{16})$, so a share (codeword) should have 16 bits. At the beginning, **A** generates uniformly at random a 448-bit secret x for **C**. The secret is then encoded into 60 16-bit symbols with a $(28, 60)$-RS code. From which, 32 parity symbols are ready to be shared. We assign exactly one share to each tag and 12 shares to **C**. In other words, without the shares from **C**, one can maximally collect 20 shares from the tags so that s/he is not able to recover the secret (even by brute force attacks). By contributing additional shares on recovering the secret, above scheme allows **C** tolerate up to 4 or 20% reading errors on scanning the tags in the case.

Similarly, at the batch level, **A** and **B** employ a $(108, 236)$-RSS scheme so that one needs to collect at least 108 shares to recover the batch secret. **A** generates uniformly at random a 1728-bit long secret y for **B**. Under the working field $GF(2^{16})$, the secret is extended into 236 16-bit symbols with a $(108, 236)$-RS code, of which 128 symbols are ready to be shared. Thus, we assign 28 shares to **B** and 100 share to the tags. With this setting, no one, except **B**, can collect more than 108 shares to successfully recover the secret. By contributing additional shares on recovering the secret, above scheme allows **B** tolerate up to 20 or 20% errors on scanning all the tags in the batch.

As an ECC algorithm requires the codewords be in an ordered sequence, we shall assign the sequence numbers on the tags explicitly. For this reason, we employ additional 16 bits in the EPC memory for the purpose of storing an ordered sequence number. This allows a quite long (up to 65536) sequence containing enough numbers of tags in a whole batch. To this end, we have used up 48 LSBs of the EPC memory and left the other 48 MSBs untouched. At the options of the adopters of our scheme, they can either retain these 48 MSBs serving as the EPC header for rough classification purpose, or fill this field with random values for privacy protection objective.

Moving forward, we work on encrypting the EPC code which is now set as 48 bits discarding the header. We hash the secret y with SHA-256 and take the lowest significant 128 bits of the output as the encryption key. Then we apply a block cipher (AES-128) in an authenticated encryption mode (*e.g.,* OCB [10]) on the EPC code with padding bits. The 128-bit encrypted and authenticated message is then stored in the user memory. Note that both the EPC memory and the user memory have similar physical and deployment characteristics (regarding the *PIN-based lock, unlock, permalock*, and *PIN-based write* operations on these memory banks) according to EPCglobal C1 G2 standard [2]. To allow **B** update the tags while pass the goods to some downstream players, our scheme requires *rewritable* EPC memory and user memory on a Gen2 tag. Such a *(re)write* operation is typically allowed in a **secured** state on interrogating a Gen2 tag, which is transitioned from an **open** state by providing the correct Access PIN. On implementing our scheme, we indicate that the 32-bit Access PIN and Kill PIN are derived from both the shared secrets x and y by **C** and **B** respectively. Also, it is not practical for **C** to access or kill a tag with the knowledge of the halves of its PINs, since guessing the other half of the Access PIN needs 2^{16} trials on the tag, which could be efficiently prevented by tag manufactures' disabling the tag when multiple false PINs are presented.

Moreover, in real-world deployment, one has to know the total number of tags R processed in a batch and the number of tags r in a case. Then s/he determines the threshold values k and K on recovering the secrets, together with the numbers of shares for the batch and cases respectively. In our running example above, the tags are formatted with $(28, 60)$-RSS scheme for a case and $(108, 236)$-RSS scheme for a batch. On choosing a proper threshold, k or K can be set as the smallest value (*e.g.,* $k = 28$, $K = 108$) that is a bit greater than the total numbers of tags in a case or batch to guarantee the recovery of secrets only with additional shares from **C** or **B**, instead of solely reading all tags in a case or batch. On the other hand, n or N could also be chosen properly to maximally tolerate reading errors (20% in our example) in a case or batch.

Last but not least, remind that we mentioned such a condition $r \geq t + \delta$ in the definition of our RSS scheme in Section 4. If there exists a relatively small number of tags in a case, our RSS scheme can adjust the relevant parameters in a resilient way. Without loss of security, we can put multiple shares on a tag or enlarge the size of a single share to minimize the value of t. For instance, we have no problem to deal with only 2 tags in a case with $(8, 16)_{32,4,2,0}$-RSS scheme with 2 shares on a tag or $(4, 8)_{64,2,2,0}$-RSS scheme with one big share on a tag. Pushing that to an extreme, for a case with only one tag, a $(2, 4)_{96,1,1,0}$-RSS scheme could be used for filling the EPC memory of the tag with a single share to achieve a maximum 96-bit security.

7 Conclusion and Future Work

In this paper, we worked on pairing supply chain parties and proposed a resilient secret sharing (RSS) scheme for distributing keying material in RFID-enabled supply chains. The scheme is proved to be secure in terms of *privacy* and *robustness*, and is resilient due to various access structures in sharing and recovering a secret. Particularly, our construction, which is based on a practical case study of "secure goods delivery", provides a set of desired security properties for batch RFID tags. Under proper parameter setting, our solution can be easily incorporated in standard RFID appliances and used in supply chain practice. Our future work is to implement our solution in real world deployments such as 3^{rd} Party Logistics in which supply chain parties are inter-connected by EPCglobal Network.

References

1. M. Bellare and P. Rogaway. Robust computational secret sharing and a unified account of classical secret-sharing goals. In *Proc. of the 14th conference on Computer and communications security*, pages 172–184, 2007.
2. EPCglobal. EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz-960 MHz, version 1.2.0., Oct. 2008.
3. A. Juels. Strengthening epc tags against cloning. In *ACM Workshop on Wireless Security – WiSe'05*, 2005.
4. A. Juels, R. Pappu, and B. Parno. Unidirectional key distribution across time and space with applications to rfid security. In *17th USENIX Security Symposium*, pages 75–90, 2008.
5. M. Langheinrich and R. Marti. Practical Minimalist Cryptography for RFID Privacy. *IEEE Systems Journal, Special Issue on RFID Technology*, 1(2):115–128, 2007.
6. Y. Li and X. Ding. Protecting RFID Communications in Supply Chains. In *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security ASIACCS'07*, pages 234–241, 2007.
7. R. J. McEliece and D.V. Sarwate. On sharing secrets and reed-solomon codes. *Communications of the ACM*, 24:583–584, 1981.
8. D. Molnar and D. Wagner. Privacy and Security in Library RFID: Issues, Practices, and Architectures. In *Conference on Computer and Communications Security – ACM CCS'04*, pages 210–219.
9. M. Ohkubo, K. Suzuki, and S. Kinoshita. Efficient Hash-Chain Based RFID Privacy Protection Scheme. In *International Conference on Ubiquitous Computing – Ubicomp'04*.
10. M. Bellare P. Rogaway and J. Black. Ocb: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security (TISSEC)*, 6(3):365–403, 2003.
11. S. Roman. *Coding and Information Theory*, volume 134 of *Graduate Texts in Mathematics*. Springer-Verlag, 1992.
12. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.