Vol.2015-CSEC-69 No.9
Vol.2015-IOT-29 No.9
2015/5/21

Secret Sharing-based Key Distribution with Dummy Tags in RFID-enabled Supply Chains

Kentaroh Toyoda[1,a)] Iwao Sasase[1,b)]

Abstract: In the RFID-enabled supply chains, it is crucial to securely convey products between parties to avoid counterfeits from being distributed. Recently, many schemes has been proposed to realize this by encrypting EPCs (Electronic Product Code) and distributing a secret key with a secret sharing scheme. However, we point out that two problems exist. The first one is that an attacker might recover the legitimate key by collecting sufficient secret shares when products are carried in the public transportation. The second one is that simply encrypting EPCs with a symmetric cipher scheme does not ensure that an encrypted EPC fits into EPC memory bank. In this paper, we first propose a secure secret key distribution scheme to solve the first problem by introducing sufficient number of dummy tags which possess a bogus secret share. Since an attacker cannot see the tags themselves from the outside of the carrying vehicle, he/she cannot distinguish between the legitimate tags and dummy tags and thus, he/she has to find out the correct key by iteratively trying each combination of secret shares. On the other hand, the party who receives products can distinguish dummy tags since they are not attached to any product. We also propose to introduce an FPE (Format Preserving Encryption) to solve the second problem. We prove that our construction is secure in both the privacy and robustness aspect. We confirm that our scheme is easily implemented with the off-the-shelf RFID reader and tags.

1. Introduction

RFID (Radio Frequency IDentification) is a fundamental technology to realize the object or product management. Especially, RFID technology is receiving much attention in supply chains to ease many complicated operations e.g., traceability, recall problem, and quality management. In RFID-enabled supply chains, a manufacturer attaches RFID tags into products and ships toward distributors or retailers. However, the counterfeit in the RFID-enabled supply chains is an open issue in the industrial and the academia due to the nature of RFID: the RFID reader can freely interrogate tags without authentication. This could be a problem once the genuine tags are interrogated by an attacker since he/she can create counterfeits that have genuine IDs. Anti counterfeit technology is an urgent demand throughout the world. For example, an OECD (Organisation for Economic Co-operation and Development) announced that the counterfeit goods in international trade could amount about $250 billion in 2007 [1]. In order to combat counterfeits, it is valid to store an encrypted EPC (Electronic Product Code) instead of a raw EPC [2].

However, in any case, the way to securely distribute the key to recipient parties is still unsolved since it is difficult to establish secure connection between the parties due to dynamic relationships in the large scale supply chains. Recently, Juels et al. proposed a novel key distribution scheme by splitting a key into multiple shares and writing them to tags with a (τ, n) secret sharing scheme [2]. The secret sharing scheme realizes that one can extract the key if he/she can obtain more than τ unique shares out of n shares [3]. Especially, any information about the key is not revealed from less than τ shares if the secret sharing has perfect secrecy. After [2] is published, many schemes were proposed to improve the secret sharing based key distribution. For instance, Cai et al. proposed a secret updating scheme to avoid an attacker from tracking paths that tags follows [4]. Lv et al. proposed an XOR based secret sharing scheme to reduce the computational cost to distribute and recover the key from shares [5].

However, we notice that there are two problems in the conventional schemes. The first one is that there is a chance that an attacker can recover the legitimate key by collecting sufficient secret shares when tag-attached products are carried in the public area. For example, an attacker can interrogate tags inside a truck while he/she is chasing abreast from it. Therefore, more secure and robust approach is necessary to deploy the secret sharing based key distribution scheme in RFID-enabled supply chains. As the second problem, when encrypting EPCs, an encrypted EPC may not fit into the EPC memory block. Since the length of encrypted message takes multiples of the block size and we cannot ensure that an encrypted EPC fits into the EPC memory bank. Considering the fact that there exist several EPC lengths (even exists user-defined variable length), the conventional schemes do not take this into consideration. Therefore, we must propose a more flexible approach to deal with any type of EPC formats.

In this paper, we propose a secret sharing based unidirectional key distribution scheme with a sufficient number of dummy tags

[1] Department of Information and Computer Science, Keio University, Yokohama, Kanagawa, 223-8522 Japan
This work is partly supported by the Grant in Aid for Scientific Research (No.26420369) from Ministry of Education, Sport, Science and Technology, Japan.
[a)] toyoda@sasase.ics.keio.ac.jp
[b)] sasase@ics.keio.ac.jp

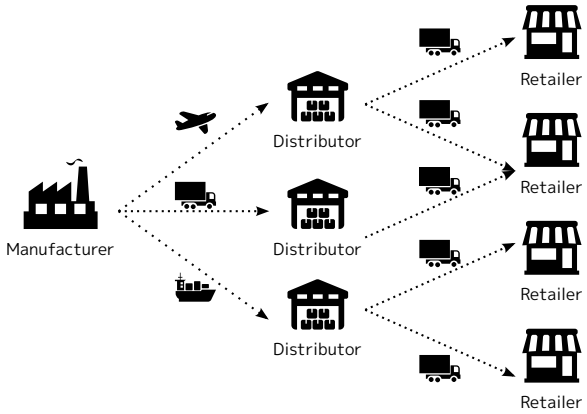


Fig. 1: Example of supply chains.

in the RFID-enabled supply chains. When tag-attached products are conveyed, we attach dummy tags which possess a bogus but plausible-looking secret share. A recipient party can put dummy tags aside when interrogating legitimate tags since they are not attached to any product. On the other hand, since an attacker cannot see tags themselves from the outside of the carrying vehicle, he/she cannot distinguish between the genuine tags and dummy ones, and thus, he/she has to find out the correct key by iteratively checking each combination of secret shares. In addition, we solve the second problem by introducing an FPE (Format Preserving Encryption) scheme [6, 7]. FPE ensures that the format of ciphertext is the same format of its plaintext. Therefore, we can deal with any type of EPC formats by specifying the format of EPC to an FPE scheme. Moreover, we argue that constructing FPE carefully provides not only the flexibility but also another security aspect, that is, by taking the possible EPC formats into consideration, any decrypted EPCs are plausible-looking even if they are decrypted with a wrong key, and thus, an attacker cannot distinguish which decrypted EPCs are correct. Our scheme is applicable to any secret sharing scheme e.g., [2, 5, 8] and thus it is flexible. In addition, we discuss the way to securely store a share in a tag against an attacker who wants to tamper shares.

We give the security analysis in the privacy and robustness aspect and clarify how many dummy tags are required to achieve sufficient security. We also measure the process time to split a key, extract a key from shares, encrypt and decrypt EPCs, and to read EPCs and shares from tags.

The rest of this paper is constructed as follows: we summarize the preliminary and related work in Section 2 and Section 3, respectively. The proposed scheme is described in Section 4. Security analysis is shown in Section 5. We evaluate our scheme in Section 6. We conclude our discussion in Section 7.

2. Preliminaries

2.1 RFID-enabled Supply Chains

RFID-based supply chains roughly consists of three parties, which are manufacturers, distributors, and retailers. Fig. 1 shows an example of RFID-enabled supply chains. Manufacturers create products, compose them in cases, and ship toward distributors. They also generate an EPC to each product, write it into an RFID tag, and attach it to a product. After distributors receive products, they decompose cases and recompose products to deliver them to retailers. Finally, retailers stock and sell them to customers. Every party equips RFID readers and interrogates tags when they arrive and leave to manage products flow.

Table 1: Example of SGTIN-96 EPC.

Field name	Example value
Header (8 bits)	00110000
Filter Value (3 bits)	010
Partition Value (3 bits)	001
Company Prefix (20-40 bits)	10010001$\cdots$
Item Reference (4-24 bits)	10011111$\cdots$
Serial Number (38 bits)	11010010$\cdots$

2.2 EPC Standard

EPC Gen2 is the de-facto standard air interface protocol for the RFID-enabled supply chains management [9]. A Gen2 system operates on UHF band (860-960 MHz), where a reader can interrogate tags from within about 10 meters. The most popular tag does not equip battery and operates by receiving continuous waves from the reader and backscatters the waves to send an EPC, TID, and some additional data. The first Gen2 specification was ratified in 2004. Then, GS1 updated a Gen2 specification as Gen2v1.2 in 2008 and then Gen2v2 in 2013 [9], respectively. In order to deal with several needs, many EPC formats e.g., SGTIN, SSCC (Serial Shipping Container Code), CPI (Component and Part Identifier), and GID, are defined by GS1 [10]. For example, SGTIN is used to assign an identifier to an item or a product and SSCC is used for containers. Table 1 shows the example of SGTIN-96. The type of EPC can be identified by the first 8 bits header. As we can see from Table 1, '00110000' indicates that the EPC type is SGTIN-96. The length of each format varies. For instance, there are two lengths for SGTIN which are 96 bits (SGTIN-96) and 198 bits (SGTIN-196). Other cases include 170 bits GRAI-170 (Global Returnable Asset Identifier) and variable length CPI.

2.3 Privacy Problem in Gen2

As we described, penetration of counterfeits goods are serious concern in the world [1]. In order to avoid counterfeits goods from being distributed in the market, it is necessary to securely hide EPCs. One solution is that a manufacturer writes encrypted EPCs into tags instead of raw EPCs and ships toward a recipient party [2]. However, the way to securely distribute a key to a recipient party is still an open issue. The most naïve approach is to construct a key management server between parties. This approach is obviously infeasible since it is difficult to decide who manages the key management server. We can also consider to simply send the password itself via secure connection over the Internet. However, it is also difficult to establish a secure connection between the parties due to dynamic relationships in the large scale supply chains. In the following section, we summarize some key distribution schemes in the RFID-enabled supply chains.

3. Related Work

Recently, the secret sharing scheme is found to be effective against a key distribution problem in RFID-enabled supply chains. To date, many researchers propose a secret-sharing based key distribution in RFID-enabled supply chains. They are classified into (1) types of secret sharing schemes being used [2, 5, 11, 12] and (2) the way to securely update contents of tags [4, 8, 13, 14].

3.1 Types of Secret Sharing Schemes Being Used

Before we summarize the unidirectional key distribution schemes in RFID-enabled supply chains, we give the fundamental of the secret sharing scheme. A. Shamir proposed the first secret sharing scheme based on the polynomial interpolation over the finite Galois Field [3]. The intuition behind this scheme is that one can determine the polynomial curve whose degree is τ when more than or equal to τ points on the curve are given while one cannot determine it when less than τ points are given. More specifically, let $f(x)$ denote a secret and a degree $\tau - 1$ curve over finite field $\mathbb{Z}_q$ where q is a prime. One can construct $f(x)$ as follows:

$$f(x) = s + a_1 x + a_2 x^2 + \cdots + a_{\tau-1} x^{\tau-1}, \qquad (1)$$

where $s \in GF(q)$ denotes a secret and $a_i \in GF(q), i \in [1, \tau - 1]$ denotes coefficients, respectively. Then, one can find n points $(x_i, f(x_i)), i \in [1, n]$ on Eq. (1) and we call them as 'shares'. If one can possess more than or equal to τ points on $f(x)$, he/she can reconstruct the intercept of $f(x)$, i.e., the secret $s = f(0)$ by Lagrange interpolation as follows:

$$f(x) = \sum_{j=1}^{\tau} L_j(x) f(x_j), \qquad (2)$$

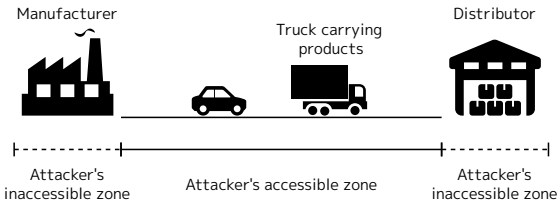
where

$$L_j(x) = \prod_{l=1, l\neq j}^{\tau} \frac{x - x_l}{x_j - x_l}. \qquad (3)$$

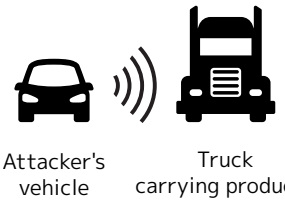
By using Shamir's secret sharing, Langheinrich and Marti proposed a secret sharing based interrogation scheme [11]. The secret information e.g., an EPC is distributed into shares and they are concatenated and written into the tag. When the reader interrogates a tag, it gradually releases the part of shares over time. After all bits are disclosed, the reader can extract the true EPC. This way avoids a hit-and-run attacker, which is an attacker who can stay close to a tag with only a limited time, from extracting a true EPC. They also proposed to distribute shares into multiple tags [12].

Juels et al. proposed a unidirectional key distribution scheme [2]. The manufacturer generates a key and splits it into shares by using a secret sharing scheme. A share is written into each tag which is attached to a product. They adopt the Reed-Solomon ECC based secret sharing scheme [15] instead of Shamir's one to reduce the size of share and to enable the party to recover the key even some shares are in erasure or error.

However, Lv et al. point out that Shamir's secret sharing scheme or Reed-Solomon ECC based scheme are computationally heavy due to the computation e.g., multiplication and division over the finite field [5]. In order to reduce the computation cost, they proposed a secret sharing scheme that only requires XOR and addition operations.



(a) Attacker's accessible zone.



(b) Example of attacker's position in the accessible zone.

Fig. 2: Attacker's accessible area.

3.2 Way to Securely Update Contents of Tags

Many researchers point out that the content of tags is unchanged throughout the supply chains and thus tags can be tracked by an attacker. Therefore, many schemes were proposed to securely update the contents of tags e.g., a written share and an encrypted EPC. Cai et al. proposed a tag-reader authentication scheme to securely update the contents of tags [4]. Although this scheme realises the secure update of tags contents, it needs modification on tags and extra hash value. Alfaro et al. proposed another approach to securely update the contents of tags by using a proactive (τ, n) secret sharing scheme [8]. Abughazalah et al. proposed to use two keys, one for cases' tags and the other for products' tags [14]. If the distributor ships tag-attached products, it newly generates the keys for cases, re-encrypts cases' EPC, and divide the cases' keys with the secret sharing scheme to avoid an attacker from tracking products.

3.3 Unsolved Problems

There are two fundamental problems that must be solved. The first one is that even though a key is split into shares and written into multiple tags, an attacker can recover the key when he/she collects sufficient (more than or equal to τ) shares. In general, the secret sharing based unidirectional key distribution is secure against a so-called "hit-and-run" attacker, which cannot interrogate sufficient shares [4, 11]. However, in reality, an attacker may be able to easily collect desired shares. For instance, when products are delivered from a manufacturer to a distributor or retailer as shown in Fig. 2, if a truck carries products through highway, an attacker may interrogate products' tags from a vehicle chasing abreast of the truck. Since EPC Gen2 tags operate on UHF, they could be interrogated from less than about 10 meters and within a few seconds. In this example, a metal container acts like a Faraday cage and so tags would be unreadable anyway from the outside. However, we can consider the case that a container is made

from non-metal material or attached tags are on-metal tags e.g., Xerafy's metal RFID tags [16]. Moreover, if an attacker can access and tamper more than or equal to $n-\tau+1$ tags, the distributor that receives products cannot recover the key without fail. Therefore, in order to deploy a unidirectional key distribution scheme in the real supply chains, we have to propose a new key distribution scheme against an attacker who can collect unbound shares rather than the limited number of shares.

The second problem is that an encrypted EPC may not fit into the EPC memory block when we naïvely encrypt it. As shown in Section **??**, there exist several EPC lengths (even user-defined variable length). Let us consider the following case: a manufacturer would like to encrypt EPCs with AES (Advanced Encryption Standard) with a 128 bits key. In this case, since the minimum block size of AES is 128 bits, the encrypted EPC takes a multiple of 128 bits and thus it cannot be written into the 96 bits or 198 bits EPC memory bank. Blowfish has many block length options though it cannot completely solve the problem yet. Therefore, since we can consider the variable types of EPC are conveyed, we must propose a more flexible approach to overcome the variable lengths problem.

4. Proposed Scheme

Here, we introduce two proposals to solve the above two problems. In order to solve the first problem, we first propose a secure unidirectional key distribution scheme by introducing a sufficient number of dummy tags. When tag-attached products are conveyed, we attach dummy tags which possess a bogus and plausible-looking secret share. Since an attacker cannot distinguish between the genuine tags and dummy ones, he/she has to find out the desired key by the brute-force manner that tries each combination of secret shares until the correct key is obtained. On the other hand, distributors or retailers who receive products can distinguish dummy tags since they are not attached to any products. In addition, we propose the way to securely store a share in a tag to avoid a robustness attacker who wants to tamper shares.

In order to solve the second problem, we leverage an FPE scheme to encrypt EPCs instead of just encrypting EPCs with a block cipher. FPE enables to encrypt a plaintext in such a way that the format of the ciphertext is same as that of the plaintext. Therefore, by introducing the FPE, an encrypted EPC fits into the EPC memory block. The idea of FPE is "*rank-encipher-unrank*" construction, that is, if we denote $L(\mathcal{F})$ is the set of strings (language) that all fit specified format $\mathcal{F}$ e.g., SGTIN-96, FPE ranks an input $x \in L(\mathcal{F})$ to an index $i \in \mathbb{Z}_{|L(\mathcal{F})|} = \{0, 1, \cdots, |L(\mathcal{F})| - 1\}$, encrypts i to yield $j \in \mathbb{Z}_{|L(\mathcal{F})|}$, and unranks j to map into $y \in L(\mathcal{F})$ [7, 17]. This way ensures that the ciphertext is an element of specified EPC format. Although FPE is originally developed to encrypt CCNs (Credit Card Numbers) or SSNs (Social Security Numbers) [6, 7, 17], we argue that FPE is also applicable to encrypt EPCs. In addition, we show that we can build a more secure construction so that an attacker cannot distinguish decrypted EPCs with wrong keys from legitimate EPCs by constructing $\mathcal{F}$ that covers all possible EPCs while does not include any unlikely candidates. We pay attention to the TDS (Tag Data Standard) specified by the GS1 [10] to construct such $\mathcal{F}$.

4.1 Assumptions

By referring the description in [2], we define our secret sharing scheme algorithm $\Pi_{\mathrm{sss}} = (\mathsf{Share}, \mathsf{Recover}, \mathsf{DummyGen})$ that operates over a key space $\mathbb{K}$.

- Share is a probabilistic algorithm that takes input $\kappa \in \mathbb{K}$ and outputs n shares $S = \{S_1, \cdots, S_n\}$ with a (τ, n)-secret sharing scheme, where $S_i \in \mathbb{S}$. On invalid input $\hat{\kappa} \in \mathbb{K}$, Share outputs n special ("undefined") symbol $\perp$.
- $\mathsf{Recover}$ is a deterministic algorithm that takes input S, τ, and n and outputs $\kappa \leftarrow \mathsf{Recover}(S, \tau, n) \in \mathbb{K} \cup \perp$, where $\perp$ is a distinguished value indicating a recovery failure.
- $\mathsf{DummyGen}$ is a probabilistic algorithm that takes input $\kappa \in \mathbb{K}$ and outputs n_D shares $\tilde{S} = \{\tilde{S}_1, \cdots, \tilde{S}_{n_D}\} \xleftarrow{U} \mathbb{S}$, where $\mathbb{S}$ denotes a share space and $\tilde{S}_i \in \mathbb{S} \ \cap \tilde{S}_i \notin \mathbb{S}_\kappa$, where $\mathbb{S}_\kappa$ denotes the share set that $\mathsf{Share}(\kappa)$ can output. On invalid input $\kappa \in \mathbb{K}$, $\mathsf{DummyGen}$ outputs n_D special ("undefined") symbol $\perp$.

We also define an FPE encryption algorithm $\Pi_{\mathrm{enc}} = (\mathsf{Enc}, \mathsf{Dec})$ as follows.

- Enc is a probabilistic algorithm that takes inputs $\kappa \in \mathbb{K}$, $x \in L(\mathcal{F})$, and a format $\mathcal{F}$. Enc outputs $y \in L(\mathcal{F})$. $\mathcal{F}$ is denoted as a regular expression. On invalid input $\hat{\kappa} \in \mathbb{K}$, $\mathcal{F}$, or x, Enc outputs a special ("undefined") symbol $\perp$.
- Dec is a deterministic algorithm that takes inputs $\kappa \in \mathbb{K}$, $y \in L(\mathcal{F})$, and a format $\mathcal{F}$. Dec outputs $x \in L(\mathcal{F})$. On invalid input $\hat{\kappa} \in \mathbb{K}$, $\mathcal{F}$, or y, Dec outputs a special ("undefined") symbol $\perp$.

We denote $\mathsf{Enc}_\kappa^{\mathcal{F}}(\cdot)$ and $\mathsf{Dec}_\kappa^{\mathcal{F}}(\cdot)$ as the encryption and decryption functions with a key κ and a format $\mathcal{F}$, respectively. For example, if one wants to encrypt a 96 bits plaintext whose ciphertext is also 96 bits, one of such $\mathcal{F}$ is $\mathcal{F} = (0|1)\{96\}$.

We assume that each party is honest and does not deviate any protocol.

4.2 Procedures

4.2.1 Operation on Manufacturers

At first, a manufacturer creates n_L products and assign a unique EPC EPC_i with an EPC format $\mathcal{F}$ and TID_i where $i \in [1, n_L]$ to each product i. A manufacturer also generates a key κ. We assume that κ is an access password which is common for tags or a generic key. A manufacturer writes $= \mathsf{Enc}_\kappa^{\mathcal{F}}(\mathrm{EPC}_i)$ and $= \mathsf{Enc}_\kappa^{\mathcal{F}}(\mathrm{TID}_i)$ and sends Lock commands in order not to be tampered by an attacker. To distribute κ to a recipient, a manufacturer splits the key κ by (τ, n_L) a secret sharing scheme $\mathsf{Share}(\kappa, \tau, n_L)$ and obtains n_L shares $S = \{S_1, S_2, \cdots, S_{n_L}\}$. Any secret sharing scheme is used, though, we adopt the Shamir's secret sharing scheme [3]. The reason is that Shamir's one is *order-invariant* while the RS ECC-based one needs an order index of share since the order of interrogated tags cannot be predicted due to the randomness of Q algorithm. The size of each share does not matter against the Shamir's one since off-the-shelf tags, e.g., [18, 19], may equip more than 512 bits memory space which can easily accommodate a 32 bits access password or a 128 bits share. Finally, A manufacturer writes them in a USER memory space and sends a Lock command to avoid an attacker from tampering them. This

way allows a recipient (and an attacker as well) to read a share from a tag while does not allow an attacker to tamper a share.

Simultaneously, a manufacturer prepares n_D dummy tags. Dummy tags are also off-the-shelf Gen2 tags. The objective of introducing dummy tags is to make it infeasible for an attacker to extract the correct key κ even if he/she collects every share in public area. The manufacturer obtains n_D bogus shares $\tilde{S} = \{\tilde{S}_1, \cdots, \tilde{S}_{n_D}\}$ with $\mathsf{DummyGen}(\kappa)$. EPCs for dummy tags are randomly chosen from the strings that satisfy the format $\mathcal{F}$. In order to avoid an attacker from distinguishing legitimate tags and dummy tags, the manufacturer writes $\tilde{S}_i$ as a file to the USER memory space. A Lock command is also sent in order not to be tampered by an attacker.

A manufacturer composes products into cases or pallets after it attaches legitimate tags to products. In order for a recipient to soon distinguish legitimate tags and dummy tags, it separately compose dummy tags without attaching to any products. Finally a manufacturer composes them to a vehicle and ships them.

4.3 Operation on Recipients

4.3.1 Key Recovering and Interrogation Tags

When products arrive, a recipient, which includes a distributor or retailer, first unpacks the products cases or pallets. It will soon detect naked RFID tags i.e., dummy tags, then puts them aside to avoid legitimate tags from being mixed with dummy ones, and interrogates only legitimate tags with a reader. After a recipient interrogates legitimate tags and obtains a share set S, it recovers the key κ with $\mathsf{Recover}(S, \tau, n_L)$. A computer attached with a reader decrypts an EPC and/or TID by $\mathrm{EPC}_i = \mathsf{Dec}_\kappa^{\mathcal{F}}(\mathsf{Enc}_\kappa^{\mathcal{F}}(\mathrm{EPC}_i))$ and/or $\mathrm{TID}_i = \mathsf{Dec}_\kappa^{\mathcal{F}}(\mathsf{Enc}_\kappa^{\mathcal{F}}(\mathrm{TID}_i))$.

Finally, a recipient updates the contents of dummy tags with $\mathsf{DummyGen}(\kappa)$. Dummy tags are reusable and thus the recipient returns them to the source party. Therefore, the process to update the contents of dummy tags is needed to avoid an attacker from knowing which interrogated tags are dummy.

4.3.2 Key Updating and Re-encryption toward Next Recipients

The procedures described above are enough for the end party of supply chains, i.e., a retailer. On the other hand, if the party further transfers products toward other distributors or retailers, it should update the contents of shares in order to avoid product tracking by an attacker. As we have seen in Section 3.2, there exist many secret updating schemes [4, 8, 14]. However they all require some modification on a tag and are not applicable for a Gen2 tag. Hence we propose a simple yet effective contents update scheme. The party generates a new key κ' and updates an access password or a key as κ' and splits into shares S' with $\mathsf{Share}(\kappa', \tau', n_L')$. The party also prepares its own dummy tags and generates bogus shares with $\mathsf{DummyGen}(\kappa')$. The party also re-encrypts EPCs and/or TIDs with a new key as $\mathrm{EPC}_i^{\mathsf{Enc}} = \mathsf{Enc}_{\kappa'}^{\mathcal{F}}(\mathrm{EPC}_i)$ and/or $\mathrm{TID}_i^{\mathsf{Enc}} = \mathsf{Enc}_{\kappa'}(\mathrm{TID}_i)$ and sends Lock commands to them. Finally, the party transfers products toward next distributors or retailers with its own dummy tags. We consider that it is not a problem to refresh a key itself because the objective is to avoid tags' contents from being exposed in the public area.

4.4 Discussion

4.4.1 How to Construct Appropriate $\mathcal{F}$

For Gen2v1 tags, our scheme encrypts EPCs with an FPE scheme. Since FPE needs a regular expression $\mathcal{F}$, we discuss how to construct an appropriate $\mathcal{F}$. We first explain the EPC formats whose length is 96 bits. If EPCs are 96 bits, the most simplest $\mathcal{F}$ is $\mathcal{F} = \mathcal{F}_{96} = (0|1)\{96\}$. This construction ensures that an encrypted EPC fits into the EPC memory bank and the problem we pointed out in Section 3.3 is solved. However, if we more tightly construct $\mathcal{F}$, we might be able to make our scheme more secure against an attacker who wants to reveal correct EPCs from decrypted ones. Let us consider the following case: an attacker knows that items in the vehicle are SGTIN-96 and tries to reveal correct EPCs by trying every key candidate. During this attack, an attacker can distinguish correct EPCs from wrong ones by observing non-compliant SGTIN-96 EPCs, e.g., '10110010$\cdots$' because the header of SGTIN-96 is defined as '00110000' by GS1 [10]. Therefore, if we specify $\mathcal{F} = \mathcal{F}_{\mathrm{SGTIN-96}} = 00110000(0|1)\{88\}$, any encrypted EPC is of the SGTIN-96 compliant format and an attacker might not be able to distinguish correct EPCs and wrong ones. This notion is easily extensible for a more complicated case that contains multiple EPC formats. When we consider tag-attached items are in containers, there might exist two EPC formats including SGTIN-96 for items and SSCC-96 for containers. Since the header of SSCC-96 is '00110001' and the other fields can take any values, we can fully cover both SGTIN-96 and SSCC-96 EPC spaces without including any non-compliant EPC candidates by specifying $\mathcal{F} = 0011000(0|1)\{89\}$. Although we can construct flexible $\mathcal{F}$, it is difficult to know how much information an attacker possesses in advance. Therefore, we recommend $\mathcal{F}$ that covers all valid 96 bits EPC formats (GDTI-96, GSRN-96, $\cdots$, GID-96, i.e., $\mathcal{F} = (00101100|00101101|\cdots|00110101)(0|1)\{88\}$. If a manufacturer ships multiple lengths EPCs such as SGTIN-96 and GRAI-170, it is necessary to prepare $\mathcal{F}$ for each length.

4.4.2 Comparison with Overinformed Attacker in [2]

Juels et al. refer to the similar attacker as an overinformed attacker in [2]. Their mention is that if an attacker sees multiple tags whose shares are generated from different keys are mixed in the warehouse of a retailer, he/she is hardly to recover each key because he/she observes too many shares. The effect of multiple products are mixed in the area is same as that of dummy tags. However such situation is limited and cannot be always made. Moreover, the effect depends on the number of other products sets and thus cannot be controlled. In contrast, our scheme can always make the situation that an attacker sees too many bogus shares by introducing dummy tags and also arbitrarily control the number of dummy tags to achieve desired security.

4.4.3 Pros and Cons

We discuss the pros and cons of our scheme. As we prove later, our scheme is more secure than the previous schemes in terms of the privacy and the robustness. For the privacy aspect, even if an attacker could collect any share from tags while they are in a public area, it is infeasible to find legitimate shares which recover the correct key. In addition, the robustness can be preserved by appropriately using the Lock command in Gen2v1.

Another advantage is that our scheme is applicable for any EPC formats specified by GS1 [10]. As we pointed out in Section 3.3, the naïve symmetric encryption adopted by the conventional schemes cannot ensure an encrypted EPC or TID fits into its memory bank. Our scheme solves this problem by leveraging FPE.

The cost of introducing dummy tags is a downside of our scheme. As we can purchase a generic RFID tag for 10-50 US cents, it costs about 100-500 USD to introduce one thousands dummy tags. Although this figure seems to be expensive, a party can reuse them again and again once it purchases them. Another downside that must be discussed is the cost and space to transferring dummy tags. However this may not be a problem since tags themselves are very light and thin and are not attached to any object.

5. Security Analysis

We prove that our scheme improves the security than previous schemes. We define the *privacy* adversary and *robustness* adversary by modeling on the work by Juels et al. [2].

Definition 1. *We call that our scheme is $(\tau, n_L, n_D, \epsilon_p, \epsilon_r)$-secure against a probabilistic polynomial time adversary who is given unbound shares.*

5.1 Privacy Adversary

A privacy adversary tries to find the key κ given unbound shares. This is weaker requirement than that of [2] since they give some limited number of shares to the privacy adversary. A privacy adversary can use the following oracles:

$O_{\mathsf{Collect}}()$: This oracle returns share sets S mixed from both legitimate and dummy tags,

$O_{\mathsf{Recover}}(S)$: This oracle returns $\tilde{\kappa}$ by inputting shares S. If $|S| < \tau$, it outputs $\perp$, and

$O_{\mathsf{Choose}}(S, \tau)$: This oracle returns by randomly choosing $S' = \{S'_1, \cdots, S'_\tau\}$ from S. If $|S| < \tau$, it returns $\perp$.

By using the above oracles, a privacy adversary tries the privacy challenge defined as follows:

Challenge $\mathbf{Cha}_{privacy}[\Pi_{\mathrm{sss}}, \mathbb{K}]$:

Input: τ, κ

Procedure:

$\hat{S} \leftarrow O_{\mathsf{Collect}}()$

$\hat{S}' \leftarrow O_{\mathsf{Choose}}(\hat{S}, \tau)$

$\tilde{\kappa} \leftarrow O_{\mathsf{Recover}}(\hat{S}')$

Output: 1 if $\kappa = \tilde{\kappa}$ otherwise 0

Claim 1. *Given our construction above, an adversary's advantage is bounded by*

$$\mathsf{Pr}[\mathbf{Cha}_{privacy}[\Pi_{sss}, \mathbb{K}] \Rightarrow 1] = \epsilon_p < \left(1 - \frac{\tau}{n_L + n_D}\right)^{n_D}.$$

Proof. An attacker can collect all shares interrogated from both legitimate and dummy tags. Since no one can distinguish legitimate tags and dummy ones, an attacker must try each combination of shares. Therefore, the probability that a privacy attacker finds the desired key κ is equivalent to the probability of choosing τ legitimate shares out of totally $(n_L + n_D)$ shares. This probability is represented as Eq. (4) and can be transformed as Eq. (5).

Table 2: Required n_D such that $\mathsf{Pr}[\mathbf{Cha}_{privacy}[\Pi_{\mathrm{sss}}, \mathbb{K}] \Rightarrow 1] \leq 2^{-\kappa}$.

(a) $|\kappa| = 32$ bits

n_L	Required n_D			
	$r_\tau = 0.7$	$r_\tau = 0.8$	$r_\tau = 0.9$	$r_\tau = 1.0$
10	153	91	58	37
100	23	17	12	7
1,000	19	15	10	4

(b) $|\kappa| = 128$ bits

n_L	Required n_D			
	$r_\tau = 0.7$	$r_\tau = 0.8$	$r_\tau = 0.9$	$r_\tau = 1.0$
10	2,140,384	397,022	102,366	32,292
100	157	110	77	47
1,000	80	60	42	19

$$\mathsf{Pr}[\mathbf{Cha}_{privacy}[\Pi_{\mathrm{sss}}, \mathbb{K}] \Rightarrow 1] = \frac{\binom{n_L}{\tau}}{\binom{n_L+n_D}{\tau}} \tag{4}$$

$$= \frac{n_L!}{(n_L - \tau)!} \cdot \frac{(n_L + n_D - \tau)!}{(n_L + n_D)!}$$

$$= \prod_{i=1}^{n_D} \frac{n_L - \tau + i}{n_L + i}$$

$$= \prod_{i=1}^{n_D} \left(1 - \frac{\tau}{n_L + i}\right) < \left(1 - \frac{\tau}{n_L + n_D}\right)^{n_D} \tag{5}$$

In Eq. (5), since $1 - \frac{\tau}{n_L+i}$, where $i \in [1, n_D]$, is always less than or equal to $1 - \frac{\tau}{n_L+n_D}$, this derives the Claim 1. □

5.2 Robustness Adversary

A robustness adversary tries to tamper some shares so that a recipient cannot recover the correct key κ.

Claim 2. *Given our construction above, a robustness adversary's advantage is bounded by*

$$\mathsf{Pr}[\mathbf{Cha}_{robustness}[\Pi_{sss}, \mathbb{K}] \Rightarrow 1] = \epsilon_r < \max\left\{\frac{\binom{n_L}{\tau}}{\binom{n_L+n_D}{\tau}}, 2^{-|\kappa|}\right\},$$

where $\max(a, b)$ *returns the bigger value between a and b.*

Proof. In our scheme, the adversary cannot tamper shares unless he/she knows the key. Therefore, the adversary's success probability is bound by the probability of identifying the correct key κ. There are two strategies for the robustness attacker to obtain κ. The first one is to recover the κ from given shares and this probability is represented by Eq. (4). The other one is to find the κ with random guessing. This probability is $2^{-|\kappa|}$. Therefore, the success probability of robustness adversary is bound by the bigger one of them. This derives the Claim 2. □

6. Evaluation

6.1 Parameterization

The question is how small the probability of Eq. (4) is. To investigate this, we clarify how many dummy tags are required to achieve lower probability than that of random guessing attack, i.e.,

$$\mathsf{Pr}[\mathbf{Cha}_{privacy}[\Pi_{\mathrm{sss}}, \mathbb{K}] \Rightarrow 1] \leq 2^{-|\kappa|}. \tag{6}$$

Since the inequality (6) is not explicitly solvable about n_D, we calculate the required (minimum) n_D that satisfies Eq. (6) through the numerical computation. Obviously, the number of required dummy tags depends on that of legitimate tags. In [2,8], they give some examples of concrete number of shipped products such as razor blades, DVDs, and pharmaceuticals. They mentioned that totally about thousands of products are initially assembled then shipped toward distributors. Distributors also disperse them toward retailers. Finally, about ten products are on the consumer side. The number of products that a party receives gradually decreases as follows: $O(10^3) \rightarrow O(10^2) \rightarrow O(10^1)$. Therefore, we variate the order of n_L from 10^1 to 10^3. Table 2 shows the required n_D versus r_τ such that $\mathsf{Pr}[\mathbf{Cha}_{privacy}[\Pi_{\mathrm{sss}}, \mathbb{K}] \Rightarrow 1] \leq 2^{-|\kappa|}$ when (a) $|\kappa| = 32$ bits (the example of an access password) and (b) $|\kappa| = 128$ bits (a generic key), respectively. Here, we denote r_τ as the ratio of τ to n_L, i.e., $r_\tau = \frac{\tau}{n_L}$. From Table 2, we can see that as the number of legitimate tags increases, the less number of dummy tags is required. Intuitively, this is because the combinations that an attacker should try get sharply increased when n_L gets large. In addition, when the large $|\kappa|$ gets large, the required n_D gets sharply increased as well when $n_L < 100$. Especially, when $|\kappa| = 32$ bits, only tens of dummy are required except for the case $r_\tau = 0.7$ and $n_L = 10$ and we can say that this is practical. On the other hand, when $|\kappa| = 128$ and $n_L < 100$, the required n_D is not acceptable for practical use. However, the case that only tens of products are shipped with a vehicle is less frequent and we can expect that a bunch of products is on a shipping vehicle. These products act like extra dummy and make an attacker more infeasible to recover the key. Finally we can see that the required n_D gets less as r_τ gets close to 1. This is because the number of correct candidates that recover the correct key gets less as r_τ approaches 1. For example, only one combination can recover the correct key κ when $r_\tau = 1$. In general, The determination of r_τ is uncleared and it is determined by the probability of erasure and/or read error of tags' contents. We argue that the number of available dummy tags might be one of the option to decide r_τ.

6.2 Process Time

When we consider the real supply chains environment, it is important to clarify the required time to split a key into shares, to extract a key by combining shares, and to encrypt/decrypt EPCs. We evaluate them with a laptop machine (MacBookPro Late 2013 that equips a dual-core Intel Core i7 2.8 GHz and a 16 GB RAM memory).

6.2.1 Required time to split and extract a key

We first measure the time to split a key with the laptop machine. We generate a 32 bits and a 128 bits key and split them into 10, 100, and 1,000 shares by using ssss version 0.5 which is a C implementation of Shamir's secret sharing [20], respectively. Table 4 shows the process time to split a key into shares. From Table 4, we can see as r_τ increases, the process time is linearly increased except for $n_L = 10$. This is because the number of variables that must be solved is $\tau = r_\tau n_L$. We can consider the reason why this fact is not true for $n_L = 10$ is that some preprocessing time e.g., reading a key from a file takes much part of the entire time. The most time consuming setting is $|\kappa| = 128$ bits, $n_L = 1,000$ and $r_\tau = 1.0$ and it takes 4.8 sec. It is acceptable when we consider real implementation. Note that dummy tags do not need any process of the Shamir's secret sharing scheme because shares to be written into dummy ones are random strings whose length is $|\kappa|$.

Table 3: Process time to extract a key from shares.

(a) $|\kappa| = 32$ bits

n_L	Process time [ms]			
	$r_\tau = 0.7$	$r_\tau = 0.8$	$r_\tau = 0.9$	$r_\tau = 1.0$
10	4.7	4.9	5.2	5.4
100	4.2×10^2	6.2×10^2	8.8×10^2	1.2×10^3
1,000	4.4×10^5	-	-	-

(b) $|\kappa| = 128$ bits

n_L	Process time [ms]			
	$r_\tau = 0.7$	$r_\tau = 0.8$	$r_\tau = 0.9$	$r_\tau = 1.0$
10	6.2	6.9	7.6	8.6
100	1.9×10^3	2.8×10^3	4.1×10^3	5.6×10^3
1,000	2.1×10^6	-	-	-

Table 4: Process time to split a key into shares.

(a) $|\kappa| = 32$ bits

n_L	Process time [ms]			
	$r_\tau = 0.7$	$r_\tau = 0.8$	$r_\tau = 0.9$	$r_\tau = 1.0$
10	4.0	4.0	4.0	4.0
100	14	16	17	18
1,000	1.0×10^3	1.4×10^3	1.4×10^3	1.7×10^3

(b) $|\kappa| = 128$ bits

n_L	Process time [ms]			
	$r_\tau = 0.7$	$r_\tau = 0.8$	$r_\tau = 0.9$	$r_\tau = 1.0$
10	4.3	4.5	4.5	4.6
100	50	57	64	74
1,000	4.8×10^3	5.4×10^3	6.0×10^3	6.7×10^3

We then measure the time to extract a key from shares with the laptop machine. We extract a 32 bits and a 128 bits key by combining 10, 100, and 1000 shares, respectively. Table 3 shows the process time to extract a key from shares. In Table 3, since our machine cannot extract a key due to memory fault when $n_L = 1,000$ and $r_\tau \geq 0.8$ we indicate them as '-'. From Table 3, we can see that the process time is exponentially increased with n_L. The process time to extract a key takes much longer time than that of splitting a key. Especially, when $|\kappa| = 128$ bits, $n_L = 1,000$, and $r_\tau = 0.7$, it takes more than 30 min to extract a key. This is not tolerant for real implementation. One of the solution to this problem is to use an XOR-based light-weight secret sharing scheme instead [5]. Note also that any party does not process against dummy tags when extracting a key and thus no additional time is required to extract a key even though the system introduces dummy tags.

6.2.2 Process time to encrypt/decrypt an EPC

We measure the time to encrypt and decrypt an EPC. We generate a 128 bits key and encrypt and decrypt an EPC whose format includes SGTIN-96, GRAI-170, and SGTIN-198. We use libFTE as a tool to encrypt/decrypt EPCs whose hash

Table 5: Process time to encrypt/decrypt an EPC.

operation	Process time [ms]		
	SGTIN-96	GRAI-170	SGTIN-198
encryption	0.24	0.31	0.49
decryption	0.29	0.38	0.60

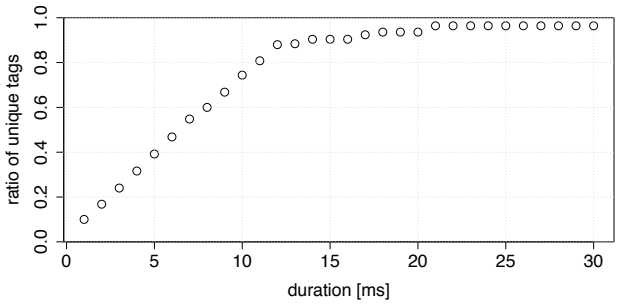


Fig. 3: Ratio of unique EPCs versus duration.

is '5c11fb3d89eeaead1c4b46f04021cd4090c61087' (the latest commit at the time of evaluation) [*1]. libFTE is an implementation of FFX (Format-preserving Feistel-based encryption) [6]. In the evaluation, we specify the input format as SGTIN-96, GRAI-170, and SGTIN-198. Table 5 shows the process time to split a key into shares. As we can see from Table 5, both encryption and decryption times linearly increase with the length of an input format. If we have 1,000 SGTIN-96 tags, it may takes a few seconds for each encryption and decryption of EPCs, respectively. From this result, we can say that the process times to encrypt and decrypt EPCs are trivial.

6.2.3 Required Time to Read EPCs and Shares

Since a unidirectional key distribution with a secret sharing scheme requires to read both EPCs and extra shares, we should show how long it takes to successfully interrogate tags. Hence we implement our scheme with off-the-shelf RFID reader and tags and measure the required time to read EPCs and shares. We use an Impinj Speedway R420 as an RFID reader, Times-7 A7030C circular polarised UHF shelf antenna, and Alien Technology ALN-9640 as tags. An RFID reader is connected with the laptop computer via Ethernet. We first generate a 128 bit key and split into shares with the (15, 20)-Sharmir's secret sharing scheme by using ssss on the laptop computer. Then, we write 96 bits EPCs and 128 bits shares into 20 tags by using Impinj Octane SDK [21]. Since ALN-9640 is a Gen2v1 compliant tag, we write a 128 bits share from the first bit in the USER memory bank. The reader antenna is located on 2 meters away from tags. We set the transmitting power of the antenna to 30 dBm which is the maximum value that our reader can set. We measure the time required to read EPCs and shares by repeating the same trials by 10 times. Fig. 3 shows the ratio of unique tags versus duration. The ratio of unique tags denotes the ratio of number of unique tags to the total tags. From Fig. 3, we can see that 30 ms is enough to read most of all 20 tags even though shares are read as well. If this is true to 1,000 tags, it will takes 15 sec to read out unique 1,000 tags. Therefore we can say that the additional reading time of shares a unidirectional key distribution is trivial and does not burden on the interrogation process.

[*1] https://github.com/kpdyer/libfte-experimental

7. Conclusion

We have proposed a secret sharing-based unidirectional key distribution scheme by introducing sufficient number of dummy tags. Since an attacker cannot see the tags inside a vehicle, he/she cannot distinguish between the legitimate tags and dummy tags even though he/she can collect all shares. Thus our scheme forces an attacker to find out the correct key by trying every combination of secret shares but the polynomial probabilistic attacker cannot find it due to the overwhelming combination of shares. We have also proposed to leverage an FPE scheme so that encrypted EPCs fits into the EPC memory bank. We prove that our scheme is secure in terms of the privacy and robustness by security analysis. In addition, we clarify that less than a few tens of dummy tags suffice when a party wants to distribute an access password and the number of legitimate products is 100-1,000. Our scheme is easily implemented with off-the-shelf tags and thus is practical for an RFID-enabled supply chains.

References

[1] Avery, P. et al.: *The economic impact of counterfeiting and piracy*, OECD Publishing (2008).
[2] Juels, A., Pappu, R. and Parno, B.: Unidirectional Key Distribution Across Time and Space with Applications to RFID Security, *USENIX Security Symposium*, pp. 75–90 (2008).
[3] Shamir, A.: How to share a secret, *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613 (1979).
[4] Cai, S., Li, T., Ma, C., Li, Y. and Deng, R. H.: Enabling secure secret updating for unidirectional key distribution in rfid-enabled supply chains, *Information and Communications Security*, Springer, pp. 150–164 (2009).
[5] Lv, C., Jia, X., Lin, J., Jing, J. and Tian, L.: An efficient group-based secret sharing scheme, *Information Security Practice and Experience*, Springer, pp. 288–301 (2011).
[6] Bellare, M., Rogaway, P. and Spies, T.: The FFX mode of operation for format-preserving encryption, *NIST proposal* (2010).
[7] Luchaup, D., Dyer, K. P., Jha, S., Ristenpart, T. and Shrimpton, T.: LibFTE: a toolkit for constructing practical, format-abiding encryption schemes, *USENIX Security Symposium*, p. 115 (2014).
[8] Alfaro, J. G., Barbeau, M., Kranakis, E. and Others: Proactive threshold cryptosystem for EPC tags, *Ad hoc & sensor wireless networks*, Vol. 12, No. 3-4, pp. 187–208 (2011).
[9] EPCglobal: UHF Class 1 Gen 2 Standard v. 2.0.0 (2013).
[10] EPCglobal: EPC Tag Data Standard (TDS) (2014).
[11] Langheinrich, M. and Marti, R.: Practical minimalist cryptography for RFID privacy, *Systems Journal, IEEE*, Vol. 1, No. 2, pp. 115–128 (2007).
[12] Langheinrich, M. and Marti, R.: RFID privacy using spatially distributed shared secrets, *Ubiquitous Computing Systems*, Springer, pp. 1–16 (2007).
[13] Li, T., Li, Y. and Wang, G.: Secure and practical key distribution for RFID-enabled supply chains, *Security and Privacy in Communication Networks*, Springer, pp. 356–372 (2012).
[14] Abughazalah, S., Markantonakis, K. and Mayes, K.: Enhancing the Key Distribution Model in the RFID-Enabled Supply Chains, *International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 871–878 (2014).
[15] McEliece, R. J. and Sarwate, D. V.: On sharing secrets and Reed-Solomon codes, *Communications of the ACM*, Vol. 24, No. 9, pp. 583–584 (1981).
[16] Xerafy: Xerafy (2014).
[17] Bellare, M., Ristenpart, T., Rogaway, P. and Stegers, T.: Format-preserving encryption, *Selected Areas in Cryptography*, pp. 295–312 (2009).
[18] Alien Technology: Higgs 3 (2014).
[19] Impinj: Monza 4 Tag Chip Datasheet (2014).
[20] point-at-infinity.org: ssss: Shamir's Secret Sharing Scheme (2014).
[21] Impinj: Octane SDK Impinj Support Portal (2014).