1-2014

Efficient Detection of Counterfeit Products in Large-scale RFID Systems Using Batch Authentication Protocols

Farzana Rahman
Marquette University, farzana.rahman@marquette.edu

Sheikh Iqbal Ahamed
Marquette University, sheikh.ahamed@marquette.edu

Efficient Detection of Counterfeit Products in Large-Scale RFID Systems Using Batch Authentication Protocols

Farzana Rahman
Department of Mathematics, Statistics, and Computer Science, Marquette University, Milwaukee, WI
Sheikh Iqbal Ahamed
Department of Mathematics, Statistics, and Computer Science, Marquette University, Milwaukee, WI

Abstract

RFID technology facilitates processing of product information, making it a promising technology for anti-counterfeiting. However, in large-scale RFID applications, such as supply chain, retail industry, pharmaceutical industry, total tag estimation and tag authentication are two major research issues. Though there are per-tag authentication protocols and probabilistic approaches for total tag estimation in RFID systems, the RFID authentication protocols are mainly per-tag-based where the reader authenticates one tag at each time. For a batch of tags, current RFID systems have to identify them and then authenticate each tag sequentially, one at a time. This increases the protocol execution time due to the large volume of authentication data. In this paper,

we propose to detect counterfeit tags in large-scale system using efficient batch authentication protocol. We propose FSA-based protocol, FTest, to meet the requirements of prompt and reliable batch authentication in large-scale RFID applications. FTest can determine the validity of a batch of tags with minimal execution time which is a major goal of large-scale RFID systems. FTest can reduce protocol execution time by ensuring that the percentage of potential counterfeit products is under the user-defined threshold. The experimental result demonstrates that FTest performs significantly better than the existing counterfeit detection approaches, for example, existing authentication techniques.

Introduction

The International Chamber of Commerce estimates that seven percent of the world trade is in counterfeit goods, with the counterfeit market being worth 500 billion USD in 2004 [1]. Counterfeiting has an impact on the rights holder, the country where counterfeiting takes place and it also causes social costs. Counterfeit, whether of clothes, medicines or CDs, cost hundreds of billions of US dollars globally every year. The effects of these crimes range from loss of company revenues to threats to public health and safety. Many companies already use anti-counterfeiting measures like holograms to confine counterfeiting and product piracy. A drawback of existing anti-counterfeiting measures is the low achievable degree of automation when checking the originality of a product. Radio frequency identification, or RFID, helps to address this problem, and provides the possibility to implement secure protection mechanisms [2].

A very good example of a large-scale system is pharmaceutical industry that needs secure and efficient anti-counterfeiting technique on everyday basis. It is one such market that is constantly fighting battles against anti-counterfeiting (which makes up to 7 % of all pharmaceutical products in the international supply chain). Since pharmaceutical products are consumed by humans, any mistake during manufacturing may cause serious harm to people's health and even lead to death. The importance of drug authenticity is obvious. In the United States, food and drug administration (FDA) has been considering the use of RFID tags to prevent counterfeit pharmaceutical products [2]. Radio frequency identification, or RFID, helps to address this problem, and provides the possibility to implement extensible, secure protection mechanisms. In fact, many RFID enabled anti-counterfeiting solutions have been already introduced in logistics, retailing, passports, etc. Because RFID has the capability of capturing and relaying data, it is what the pharmaceutical industry is looking toward to improve quality, reduce costs, and improve patient safety.

The RFID tags are typically low-cost and pervasive devices, being attached to products or objects to enable the identification of those objects. A tag has small microchip and an antenna on board. The reader can collect the ids of tags via RF signals, without the need of keeping line of sight. As an effective automatic processing measure, RFID offers several attractive features over the barcode, such as non-optical proximity, interactive communication, rewritable ability, etc.

A common technique of RFID enabled anti-counterfeiting is that the manufacture stores a serial number K (or termed as *secret key*) for each tag. The secret key is also stored in the central authentication server. During authentication, an RFID reader challenges a tag for its validity and the tag replies with its encrypted or hashed serial key. This encrypted message is passed to the server for validity checking. If the serial number is valid, the product to which the tag is attached is declared as genuine. During this process, however, an adversary can eavesdrop in between the channel and the learned information can be used to create a counterfeit tag. To address this issue, many efficient and private authentication protocols have been proposed in literature. Weis et al. [3] propose an authentication scheme based on Hash lock. The search complexity of Hash lock is $O(N)$, where N is the total number of tags in the system. To improve the search efficiency, tree-based approaches [4–6] convert the verification process to a depth-first-search in a key tree to reduce the search complexity to $O(\log N)$. However, the reader still needs much longer time to authenticate products in large supply chain.

To solve the problem of RFID-based counterfeit detection, in this paper, we propose to authenticate tags in batches. However, if we apply straightforward private authentication technique, the protocol execution time will still be very high. If we consider this problem from a different perspective, we see that it is not always necessary to ensure the genuineness of every single product in a batch. In fact, even in the genuine products, there can be some products that are shipped as defectives from the manufacture. So it is acceptable if we guarantee the percentage of counterfeit products is sufficiently small. At this point, we summarize our contributions in this paper

- We propose to verify the validity of a batch of tags using statistical inference-based sampling in a protocol named GTest. However, it is not efficient since it needs high execution time and large volume of authentication data.
- To solve the problems of efficient batch authentication, we propose FSA-based authentication protocol (FTest) that uses a variation of framed slotted Aloha [7] technique.
- To compare the performance of GTest and FTest protocols with a per-tag authentication protocol, we measure their execution time in a simulated RFID environment.

The contributions of this paper are partially presented in [31]. The rest of this paper is organized as follows: Sect. 2 describes our major motivation. We introduce preliminary knowledge about different authentication and anti-collision protocols in Sect. 3. Our system model and assumptions are mentioned in Sect. 4. In Sects. 5 and 6, we describe the sampling-based GTest and FSA-based FTest batch authentication protocols, respectively. We discuss the security analysis of FTest protocol in Sect. 7. We examine the performance of FTest and GTest protocol in Sect. 8. This section is followed by the relevant related work in Sect. 9. Finally, we conclude the paper with some future works in Sect. 10.

Motivation

When detecting product counterfeiting in large systems, existing approaches can be impractical since tags needs to be authenticated one at a time. It may seem at first that the problem of counterfeit tag detection can be solved by using any RFID tag identification or authentication protocol, simply by allowing the reader to interact with the tags of the batch. However, this deterministic process will be very time consuming since reader needs to authenticate each and every tag of the batch to determine its validity. The situation will be even worse if there are large numbers of tags in the system. Such low authentication efficiency is unacceptable in practice especially in large-scale supply chain. Therefore, we need batch authentication not only to increase efficiency but also to prevent counterfeiting. In this paper, we propose a batch authentication protocol that is scalable, efficient as well as able to prevent product counterfeiting within a user-defined tolerance level. Due to this protocol, mass authentication along the supply chain can be possible, and the cost of maintaining integrity of supply chains can be significantly reduced. Eventually it may increase health security, social awareness, and global trading concerns.

Background

In this section, we discuss how framed slotted Aloha (FSA) and tree-based authentication protocol work.

Tree-based authentication protocol

In tree-based hash protocol [8–10, 25], the tags are organized in a secret key tree where each tag is assigned to a leaf of the tree. Secret keys are associated with each branch of the tree. Each tag (each leaf) receives all the secret keys along the path from the root to itself. If the tree has *L* levels, each tag stores *L* keys. The key tree is a

balanced tree, therefore, if the branching factor is α, the $\log_\alpha N$ will be equal to L. Each tag has only one key that is not shared with any other tag of the system. Figure 1 shows a balanced key tree with $N = 8$ and $\alpha = 2$.

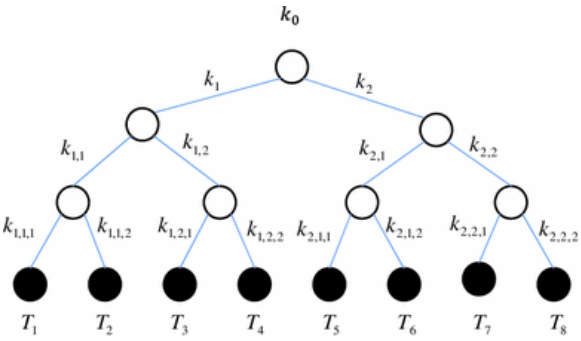


Fig. 1 A secret key tree for the tree-based hash protocol with $N = 8$ and $\alpha = 2$

According to this protocol, the reader queries a tag with a nonce n_r. Upon the reception of the nonce from the reader, the tag replies to the reader with

$$h(k_0, n_r), h(k_{l_1}, n_r), h(k_{l_1,l_2}, n_r), \ldots, h(k_{l_1,l_2,\ldots,l_L}, n_r),$$

where each $l_i \in \{1, \ldots, \alpha\}$, $1 \le i \le L$ and $h(\cdot)$ is a hash function. After receiving the response, the reader first finds a match with the first hash value of the response by hashing with all the keys of level 1. Whenever the reader obtains a match, the reader starts to search for the second hash value of the response by hashing with all the keys at the next level of the sub-tree rooted at the node where the reader has found the match. The reader repeats this step until it reaches a leaf. Thus, the reader's complexity is reduced to $O(\log_\alpha N)$. In the worst case, the reader has to search with all the keys at each level of the tree and the complexity becomes $\alpha(\log_\alpha N)$.

The major drawback of this approach is each tag must transfer 4 hash values to the reader at each authentication. As we discussed before, such a large volume of data is a major bottleneck preventing us from accelerating the batch authentication.

Framed slotted Aloha

Aloha-based protocols are important because they reduce the probability of occurrence of tag collision. In case of pure slotted Aloha, tags select their response time arbitrarily. In slotted Aloha [7], tags select the timeslot randomly and reply at the beginning of the timeslot to avoid overlapping of transmissions. Framed slotted Aloha (FSA) algorithm can be used to identify a batch of tags. The protocol uses a fixed frame size and does not change the size during the process of tag identification. In FSA, the reader offers information to the tags about the frame size (f) and the random number (r) which is used for a tag to select a slot number in the frame. Each uses a hash function $h(x)$, which is used to choose the slot number. After receiving f, each tag selects $h(id \oplus r) \bmod f$, as its slot number. The reader then sequentially scans every slot in the frame. In each slot, if a tag's slot number equals zero, it will send its *id* to the server immediately. Otherwise, the tag reduces its slot number by one. Since a tag cannot sense the signals replied from other tags, there are three types of slots from the reader's perspective—slots with no reply, single reply, or multiple replies. We define these slots as empty slot, single-reply slot, or collision slot, respectively. Slots can also be characterized as *multi-bit response slots* and *single-bit response slots*. Since the frame size of FSA is fixed, its implementation is simple but it exhibits low efficiency of tag identification. For example, if there are too many tags, no tag may be identified since the slots experience high collision. On the contrary, many slots wasted if the number of tags is small. Our FTest protocol is partially dependent on the concept of FSA.

System model and problem formulation

Problem definition

In our system, we assume that each object is attached with an RFID tag that has a unique *id* (e.g., *secret key*). We define the set of tags as population. These tags are divided into batches or groups of equal size. Suppose, *N* is the total number of tags in the system and τ is the number of groups. So, the group size is $n = \frac{N}{\tau}$. The number of tags in a batch, *n*, is known in advance. We define a batch of tags as valid if no counterfeit tag is detected, otherwise this batch is considered as invalid. In our system, each batch is associated with a unique key that we refer to as a *group key*. In addition to each tag's own secret key id_i, every tag shares this group key with other members of the given group. Figure 2 shows the group organization of the tags where $N = 8$ and $\tau = 4$. The k_i's are the group keys, where $1 \le i \le \tau$.

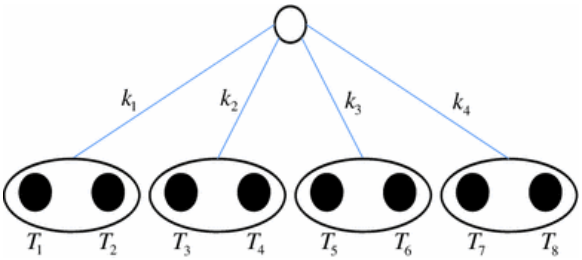


Fig. 2 Group organization of the tags for the batch authentication protocol, with *N* = 8 and τ = 4

Architecture of the system

There are mainly four components in the system:

Issuer: The issuer initializes each tag during the deployment and also authorizes the reader access to the tags. We can think of the issuer as a certificate authority (*CA*).

RFID Tag: Each RFID tag is denoted as *T*. The issuer assigns a unique key id_i and a group key *k* to the *i*th tag T_i of the system. The use of group key will be explained later.

Reader: A reader (*R*) connects to the authentication server through a high-speed network. In this paper, we assume the communication channel between the reader and the backend server is secured. The reader receives all the secret information by the issuer during the deployment.

Server: The authentication server maintains all the group keys and *N* secret keys corresponding to *N* tags in the database. The server also knows which tag belongs to which group by maintaining a database like Table 1. The server has powerful computing capability.

Table 1 Database structure of the authentication server

Group key	Tag key	Tag name
k_1	id_1	T_1
	id_2	T_2
	...	...
	id_i	T_i
...	...	...
k_n	id_{j+1}	T_{j+1}
	...	...
	id_N	T_N

Preliminaries and assumptions

We assume that the reader and all the tags in the system have the knowledge of XOR operation and $h(\cdot)$, an irreversible one-way hash function to protect the integrity of the message. The outputs of $h(\cdot)$ cannot be linked back to its input so that an adversary cannot link back the tag *id*s. There are many efficient hash functions in the literature. The hash $h(\cdot)$ does not need to be a cryptographic hash function. In order to keep the tag's hardware simple a cyclic redundancy check (CRC) function which is already found in existing RFID tags can be used as $h(\cdot)$. Communication between the reader and the tags is time-slotted. The synchronization between the clocks of the tags and reader is done by "start" signal of the reader. The reader uses a 'slot start' command to start a slot. Our protocols are request-response-based protocol, in which the reader issues a request in a time slot and then zero, one or more tags respond in the subsequent time slots. We assume that RFID reader is able to distinguish the three types of slots mentioned earlier. All the notations related to our system is shown in Table 2.

Table 2 Notations

Symbol	Meaning
T^*	Set of RFID tags
R	RFID Reader
N	Number of tags in the population
n	Number of tags in a batch
τ	Number of batch in the system
$h(\cdot)$	One-way hash function
Δ	Counterfeit threshold
n_s	Sampling size

Protocol goal

The goal of a server is to accurately and efficiently determine the validity of a batch of tags. It may seem at first that the problem of counterfeit tag detection can be solved by using any RFID tag identification or authentication protocol, simply by allowing the reader to interact with the tags of the batch. However, this deterministic process will be very time consuming since reader needs to authenticate each and every tag of the batch to determine its validity. The situation will be even worse if there are large numbers of tags in the system. In this paper, we design a probabilistic protocol to solve the batch authentication problem by using a variation of FSA-based tag detection algorithm. We call our protocol probabilistic since FSA itself is a probabilistic protocol. It will provide a provable probabilistic guarantee for valid batches of tags ensuring the percentage of potential counterfeit products is less than counterfeit threshold (Δ). Δ is a system parameter defined by the user in advance. We guarantee a batch is valid if there are no more than $n \times \Delta$ counterfeit tags in the batch. Note that it does not mean that the batch will be declared as valid if the number of counterfeit tags is lower than $n \times \Delta$. Even if there is only one counterfeit tag in the batch, it will still be declared as invalid.

Group test (GTest) batch authentication

In this section, we present a batch authentication process called group test (GTest) that uses statistical inference to determine the validity of a batch of tags. The concept of GTest is to reduce the cost of detecting counterfeits in large populations which is believed to contain a small proportion of defectives by drawing sample from the large population randomly. If GTest protocol is applied to products of batches in a large supply chain, then there may be no interest in knowing which products are defective. The purpose may instead be to accept or reject the batch or to estimate the number of counterfeit products it contains. Therefore, it is useful to know the probability distribution of the number of counterfeit samples.

GTest protocol design

GTest protocol operates in two phases—(1) *Group identification* and (2) *authentication*. In the first phase, the reader queries the tags with a nonce n_r. The tags, then, replies the following encrypted message with probability 0.5

$$h(k_i \| n_r)$$

here, k_i is the group key in which the tag belongs. Now the reader tries all the group keys to decrypt this message. If the reader finds the right group key that correctly decrypts the message, then the reader can learn the identification of all the tags corresponding to that group by online querying the database of the server. This process of tag identification is much efficient than per-tag-based identification since the reader do not need to query each individual tag of the batch. The reader will, then, start the authentication process by randomly selecting *m* tags as samples and collecting the authentication data from them. Next the reader forwards these data to the server. GTest declares this batch of tags as invalid if the server can detect one invalid or counterfeit tag.

Protocol analysis

According to the statistical inference based on sampling, we can estimate the proportion of individual samples that are defective when they have been taken at random from a large population. If *n* individual samples are combined τ at a time to give *m* pooled samples, then the number of counterfeit pooled samples follows approximately the binomial distribution $B(m, \delta_t)$, where

$$\delta_t = 1 - (1 - \delta)^t = \text{probability that a pooled sample is positive}$$

here, δ is the probability that an individual sample chosen at random from the entire population is defective. Suppose, that our complete population of tags (i.e., *N* tags) contains n_c counterfeit tags. Therefore,

$$\delta = n_c / N$$

A pooled sample is assumed to be invalid if and only if it includes at least one individual counterfeit sample. Then, the probability that exactly η of the pooled samples may be invalid is given by

$$f(\eta | n_c) = \binom{m}{n} \sum_{i=0}^{\eta} (-1)^i \binom{\eta}{i} \binom{N-\eta_c}{n-(\eta-i)t} / \binom{N}{n-(\eta-i)t}$$

(1)

where, $\max(0, m - \frac{(N-n_c)}{t}) \leq \eta \leq \min(m, n_c)$

In Eq. 1, when η is zero, we derive—

$$f(\eta|\eta_c) = \binom{N-n_c}{n} / \binom{N}{n}$$

(2)

Equation 2 refers to the hypergeometric probability since the absence of counterfeit among the *n* individual samples is equivalent to the absence of positive pooled samples.

Now, we know that the number of counterfeit tags follows the hyper-geometric distribution; we define random variable *X* to refer to the number of counterfeit products in a batch. Suppose, in a batch with *n* tags, we sample n_s at a time. Then, the pdf of *X* will be:

$$f(X|n\Delta) = \frac{\binom{n\Delta}{X}\binom{n(1-\Delta)}{n_s - X}}{\binom{n}{n_s}}$$

here, $E[X] = n_s \times \Delta$ However, using GTest protocol, if we want to identify the validity of a batch, reader needs to read a high amount of data. For example, with n = 100,000, n_s = 1,000 and Δ = 0 % (meaning that every counterfeit tags need to be detected), reader needs to read $1{,}000 \times 20 \times \log^{1{,}00000}_2 = 3.1$ MB of data which will take high response time. So, to decrease the protocol response time, we propose a more efficient protocol in the next section.

FSA-based (FTest) batch authentication

In this section, we propose FTest protocol which is dependent on a variation of *frame slotted Aloha technique*. We consider an RFID reader *R*, and a population of *N* RFID tags denoted as *T**. Table 3 summarizes the notations for FTest protocol.

Table 3 Notations for FTest protocol

Symbol	Meaning
SP	Slot position within frame
RV	Response vector generated by the reader with the replies of tags
RV_s	Response vector generated by the server
rem	Set of tags that are removed from the authentication initialization phase to reduce collision slot

FTest protocol design

FTest has three phases: (1) *Group identification*, (2) *Authentication initialization,* and (3) *Counterfeit detection*. The group identification phase is similar to the one mentioned in GTest batch authentication protocol. The other two phases are discussed next. The entire process is illustrated in Fig. 3.

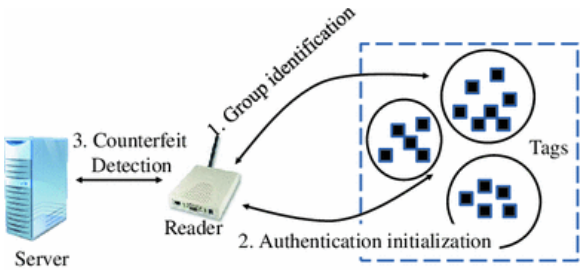


Fig. 3 Authentication process of FTest protocol

Authentication initialization

After identifying a group of tags using the group identification mechanism mentioned in GTest protocol, the authentication phase is initialized. Reader simply starts the authentication by sending "*start authentication*" command to the server and by receiving a frame size *f* and random number *r*. The reader broadcasts the *f* and *r* received from the server in the first step. The frame consists of *f* short-response time slots right after the request. Each tag uses the random number *r* and its key (*id*) to hash to a slot position, *SP*, between [1, *f*] where

$$SP = h(id, r) mod f$$

The tag transmits a short response at that slot (ex. 1 bit). Therefore, the time duration of all slots in our approaches is very short. After the frame ends, the reader abstracts the responses in the frame as a response vector (*RV*). *RV* is a vector in which each element is related to a slot in the frame. There are three types of elements in an *RV*—0, 1, and collision, representing empty slot, single-reply slot, and collided slot, respectively.

We modify the slot picking behavior so that instead of having a tag, pick a slot and return its *id*, we let the tag reply with 1 bit of information signifying that the tag has chosen that slot. In other words, instead of the reader receiving

$$\{\dots|id1|0|\dots|collision|0|1|\dots\},$$

where 0 indicates no tag has picked that slot to reply, and collision denotes multiple tags trying to reply in the same slot, the reader will receive $\{\dots|1|0|\dots|randombits|0|1|\dots\}$.

This is more efficient since the tag *id* is much longer than the bits transmitted. However, this approach is still inefficient because the information carried in the collision slot is totally unused. To utilize the collision slot, we turn it into a single-reply slot by removing one of the two tags from this phase. Suppose two tags T_i and T_j map to the same slot position. We remove the tag (T_i) from this phase so that it does not transmit any short response. Hence, another tag (T_j) corresponding to this slot has a singleton slot which allows it to be authenticated. This situation can be made more efficient by turning each *p*–collision (*p* tags mapped into one slot) slot into a singleton slot by randomly removing *p*–1 tags.

All the tags that are removed using this process are instructed to keep silent in this phase and they are authenticated in the next phase. We refer to this set of tags as rem. In counterfeit detection phase, the reader authenticates those tags after another to verify their validity. The authentication protocol is shown in Fig. 4, 5, and 6. Each tag in the set executes Algorithm 1 (see Fig. 4) independently. The reader executes Algorithm 2 (see Figs. 5) to generate the *RV* and return it to the server.

Algorithm 1: *Algorithm executed by RFID tags*

Receive (f, r) from R

for each tag T_i (where i = 1 to N)

    compute $SP_i = h(id_i, r)$ mod f

end

while R broadcasts Slot Position (SP)

    if $(SP == SP_i)$

        return 1 in $RV[SP_i]$

end

Fig. 4 Algorithm executed by tags in FTest protocol

Algorithm 2: *Algorithm executed by reader R*

Define RV of length f
Initialize all entries of RV to 0
for Slot Position $SP = 1$ *to* f **do**
  Broadcast SP and listen for reply
  if $(no\ reply)$
    continue
  else if $(reply! = collision)$
    $RV[SP] = 1$
  else $RV[SP] = collision$
end
return RV to the server

Fig. 5 Algorithm executed by reader in FTest protocol

Algorithm 3: *Counterfeit Detection*

assign $total\_counterfeit\_tag$ = number of counterfeit tag detected during per tag authentication
while $(total\_counterfeit\_tag/n < \Delta |$ entire RV checked$)$
  if $(RV[i] == 0\ \&\ RV_s[i] == 0)$ *continue*
  else if $(RV[i] == 1\ \&\ RV_s[i] == 0)$
    assert *counterfeit detected*
    $total\_counterfeit\_tag$ ++
  else if $(RV[i] == collision\ \&\ RV_s[i] == 1)$
    assert *counterfeit detected*
  else continue
end

Fig. 6 Counterfeit detection process in FTest Protocol

Counterfeit detection

In this phase, first the server starts the detection process by challenging the tags belonging to rem with a nonce n_r. The tags, then, replies the following encrypted message: $h(id \parallel n\_r)$.

The server considers those tags as valid for which it can find legitimate ids able to generate the corresponding hash values. Tags that cannot authenticate themselves are considered as counterfeit tags. After this per-tag authentication process is over, server starts to verify the validity of *RV.* Since the server knows all the keys of the tags corresponding to that batch, it can use those keys for reconstructing the *RV*. The server knows the locations of the empty, singleton, and collision slots. If such reconstructed response vector exists, which we name as RV_s, the server deterministically accepts the batch of tags as valid. Otherwise, the batch is invalid. Because a counterfeit tag has no valid key, its corresponding reply is not expected. So if a slot is supposed to be empty but the server finds it singleton, then the server asserts the existence of counterfeit tag. If a slot is supposed to be singleton, but the server finds a collision, then at most one tag of that slot is valid and it is also an indication of the existence of counterfeit tag. Otherwise, the server goes to the next slot position. After the checking ends, if there is no counterfeit tag detected, the batch will be accepted as valid.

Since our goal is to declare a batch invalid if the percentage of counterfeit tags exceeds counterfeit threshold Δ, we incorporate that parameter in our counterfeit detection process. This detection process will not continue if the number of total counterfeit tags in the batch to the number of total tags in the batch is greater than Δ. This

will significantly reduce the number of rounds in the counterfeit detection protocol since the entire *RV* does not need to be checked. It will also reduce the response time of the entire protocol. For example, suppose *n* = 1,000, number of counterfeit tag detected during per-tag authentication is 35. Number of counterfeit tags detected during the first 70 rounds of counterfeit detection protocol is 15. Then,

$$Total\_counterfeit\_tag/n = \frac{50}{100} = 0.05$$

If Δ = 5 %, then the counterfeit detection protocol will stop after 70th round. The complete counterfeit detection algorithm is shown in Fig. 5. With *n* = 100,000, n_s = 1,000 and Δ = 0 %, FTest reads 0.03 MB of data per batch.

Protocol analysis

- In FTest protocol, we assume that all tags in a batch, both legitimate and counterfeit, will reply at least once in the frame. However, the counterfeit tags may reply more than once to introduce more collision and we name this type of attack as "*collision attack*". Additionally, the counterfeit tags may not reply at all to hide their identity and we name this attack as "*concealing attack*". It is very hard to defend against concealing attack and it is out of the scope of this paper. We can identify the collision attack by comparing the *RV* (response vector returned by the reader) and RV_s (response vector generated by the server for genuine tags only). There can be following types of distinguishable situations that indicate the existence of collision attack:
- When $RV_s[i] = 0$ and $RV[i] = 1$, there should be no genuine tags replying in this slot. But the result shows one tag has chosen this slot. So, this tag must be a counterfeit.
- When $RV_s[i] = 0$ and $RV[i]$, there should be no genuine tags replying in this slot. But the result shows more than one tags of this batch has chosen this slot. This also ensures that there are counterfeit tags in the system.

When $RV_s[i] = 1$ and $RV[i]$ = collision, there should be only one genuine tag. But the result shows more than one tags has chosen this slot. It implies that at most one tag replied in this slot is genuine and the rest are counterfeit.

Security analysis of FTest

Attack model

The goal of an adversary in our system is to install counterfeit tags in the system. Evidently, this fake tag can let a fake object to be identified as an authentic one. In this paper, an adversary is denoted as $\hat{A}$. We assume $\hat{A}$ is an active adversary who has full control over the entire communications channel between the tags and the reader. She can eavesdrop in between the channel and can use the learned information to create counterfeit tags and install in the system. Each counterfeit tag is denoted as $\hat{T}$. These counterfeit tags are just like genuine tags except their secret key, and group key is issued by the adversary. The attacker does not have any control over any other component of the system. However, the attacker may try to violate privacy and track each and individual genuine tags of the system. Our assumptions for this paper also include that genuine tags and reader cannot be compromised by the attacker. In our system, the following oracle-like construction exists:

$\mathcal{O}_{Eavesdrop}(\boldsymbol{R}, \boldsymbol{T}, \boldsymbol{t})$: The adversary eavesdrops within a channel to listen to the communication protocol session at time *t* between reader R and one of its communicating tag *T*.

$\mathcal{O}_{\boldsymbol{Impersonate_R}}(\boldsymbol{R},\boldsymbol{T},\boldsymbol{M},\boldsymbol{t})$: The adversary impersonates a reader *R* in a protocol session at time *t* and sends a message *M* to the tag *T*.

$\mathcal{O}_{\boldsymbol{Impersonate_{WT}}}(\boldsymbol{R},\boldsymbol{T},\boldsymbol{M},\boldsymbol{t})$: The counterfeit tag *T* impersonates a genuine tag in a protocol session at time *t* and sends a message *M* to the reader *R*.

$\mathcal{O}_{\boldsymbol{Query}}(\boldsymbol{T},\boldsymbol{t})$: The adversary queries a tag *T* to learn information during a communication session at time *t*.

$\mathcal{O}_{\boldsymbol{Receive}}(\boldsymbol{U},\boldsymbol{M},\boldsymbol{t})$: The adversary receives a message *M* from an entity *U* (e.g.,, either *T* or *R*) during the execution of protocol session at time *t.*

Security analysis

In this section, we analyze our proposed authentication protocol against different types of attacks. For every attack, we first describe how the attack is performed by an adversary. Then, how our protocol protects against the attack is explained. *R* and T_i are referred to as a legitimate reader and legitimate tag. Each attack and defend, as a whole, have three phases:

Phase 1

Learning phase: This phase represents pre-attack preparations. Adversary, A^A^ uses non-destructive oracles such as $\mathcal{O}_{Eavesdrop}(R,WT,t)$, $\mathcal{O}_{Query}(WT,t)$, $\mathcal{O}_{Impersonate_R}(R,T,M,t)$, and $\mathcal{O}_{Receive}(U,M,t)$ on a set of target tags and reader to learn information related to them.

Phase 2

Attacking phase: $\hat{A}$ starts to attack by creating counterfeit tag ($\hat{T}$) and installing them in the system.

Phase 3

Defend Phase: Our protocol is designed in such a way so that it can defend against majority of the attacks performed by the fake tag $\hat{T}$.

Since the defend phase is different for each attack, we discuss this phase for following attacks:

1. *Collision Attack*

Learning and attack phase:

Under this attack, $\hat{A}$ queries a set of valid tags with different (*f*, *r*) using $\mathcal{O}_{\boldsymbol{Impersonate_R}}(\boldsymbol{R},\boldsymbol{T},\boldsymbol{M},\boldsymbol{t})$ oracle to collect replies from the genuine tags. Using this learned information, she creates fake tags and installs them in the system. From then on, the fake tags will try to attack the system with their responses.

Defend phase:

In collision attack, a counterfeit tag emits replies in multiple slots for disturbing the distribution of slots in the *RV*. In fact, our approach is very immune to such attack, since generating more meaningless replies is equivalent to increasing the ratio of counterfeit tags, which helps to increase the probability of detecting counterfeit tags.

2. *Privacy violation attack*

Learning phase:

Under this attack, $\hat{A}$ repeatedly queries T_i with different (*f*, *r*) using $\mathcal{O}_{Impersonate_R}(\boldsymbol{R}, \boldsymbol{T}, \boldsymbol{M}, \boldsymbol{t})$ oracle to collect replies from the existing tags.

Attacking phase:

$\hat{A}$ execute $\mathcal{O}_{Receive}(\boldsymbol{U}, \boldsymbol{M}, \boldsymbol{t})$ oracle to learn replies from the tags and create a response vector. The goal of the attacker is to learn the *id*s of different tags.

Defend Phase:

Our protocol can preserve the privacy of individual RFID tag since none of the tags reply their *id*. Therefore, the adversary cannot infer the *id*s from the replies of the tags.

3. *Tracking attack*

Learning phase:

Here, $\hat{A}$ tries to track T_i over time. $\hat{A}$ succeeds if it can distinguish WT_i from other tags over time.

Attacking phase:

Under this attack, $\hat{A}$ repeatedly queries T_i with different (*f*, *r*) using oracle $\mathcal{O}_{Query}(T_i, t)$ to learn about the slot picking behavior of the tags. Then, the adversary executes oracle $\mathcal{O}_{Receive}(U, M, t)$ to receive replies from the tags. The goal of the attacker is to get a consistent reply that may become a signature of T_i.

Defend Phase:

FTest protocol is resistant against tracking. Let an adversary $\hat{A}$ eavesdrops on the transaction between a reader *R* and the genuine tags. So $\hat{A}$ knows the queries and replies but based on this information the adversary cannot compute reply for any random query. The adversary can certainly be sure that a communication has taken place. However, it cannot figure out which tag replied in which slot since it do not have *id*s of the tags. Moreover, the slot picking behavior of the tags changes with the change of *f* and *r*. Therefore, the outputs of all the tags seems to be pure random to the adversary $\hat{A}$.

4. *Eavesdropping attack*

Learning phase:

$\hat{A}$ executes the oracle $\mathcal{O}_{Eavesdrop}(R, T_i, t)$ and later uses this information to launch different attacks (ex. replay attack).

Attacking phase:

$\hat{A}$ learns every information exchanged between *R* and T_i. The goal of $\hat{A}$ is to use the data to impersonate a fake tag.

Defend phase:

Our protocol is powerful against this attack. In our protocol $\hat{A}$ will not be able to find out the expected reply of the tags. In each pass, all tags will pick a different slot based on the random number sent by the reader. $\hat{A}$ can

only observe the communication but it cannot link the outputs of the two parties. It cannot even launch replay attack by replaying previous messages since the slot picking nature of the tags changes with *f* and *r*.

Evaluation

We evaluate the efficiency of GTest and FTest protocol. Our comparison is based on the metric–*execution time*. To compare the performance of both protocols, we also simulate a per-tag authentication (PTA) protocol to identify the validity of batches. PTA is a deterministic approach, which authenticates all tags to detect the validity of a batch. The accuracy of PTA is certainly 100 % but its efficiency is very low. There are plenty of per-tag authentication protocols in literature [4–6, 11–13, 25, 29].

We use AnonPri [6], a group-based authentication protocol as PTA. In our simulation, the authentication server is implemented on a high performance Dell PC. We use java for protocol simulation where we use SHA-1 as the hash function (returning 160 bits) in all three protocols. We also use MySQL to store secret keys for the simulated tags. In our simulated RFID environment, we have considered two systems with $N = 2^{16}$, τ = 8, 16, 32, 64 and $N = 2^{20}$, τ = 512, 256, 128, 64. We deliberately introduce 1–4 % randomly generated counterfeits into the dataset. We have run the simulation for 100 times and reported the average.

Execution time metric determines the time required for interaction between the reader and tags. This metric tells us the processing time of a protocol to determine the validity of tags with Δ = 0 %. It means that we need to identify all the counterfeit tags in each batch. However, Δ is a system parameter and it can also be changed to observe the change of execution time. Since every bit almost consumes the same transmission time which equals 25 μs [14] on average, we measure the execution time by multiplying the size of transmitted data (in bits) with 25 μs.

For all protocols, we consider the tags uses SHA-1 hash function. Therefore, in GTest protocol, the length of data replied by tags is 160 bits [15]. The total size of data used for group identification equals 160 × *n*/2 (since tags will reply with probably 0.5). Now to verify a batch with *n* tags, suppose that n_s tags are sampled and the length of random numbers equals 160 bits. The size of authentication phase will equal (160 + 160) × n_s bits, since reader will challenge with a random number (160 bits) and tags will reply with their hashed response (160 bits). Therefore, the total data size of GTest protocol will be:

$$d\_\text{size}_{\text{GTest}} = \left(160 \times \frac{n}{2} + (160 + 160) \times n_s\right)\text{bits}$$

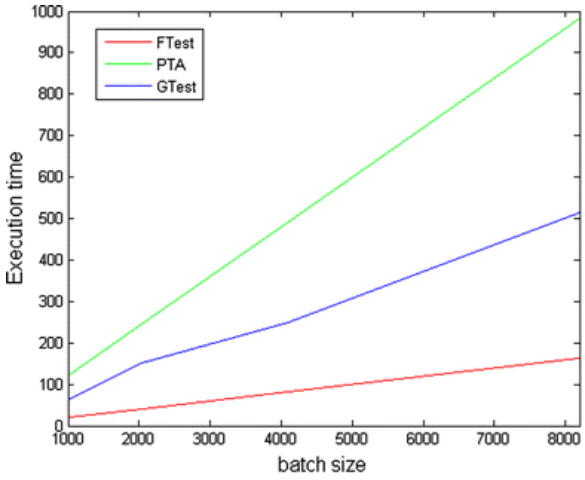
For PTA protocol, the reader needs to check all the group keys and this has to be done for all the tags of the batch. Therefore, the data size will be

$$d\_\text{size}_{\text{PTA}} = (160 \times n + (160 + 160) \times n)\text{bits}$$

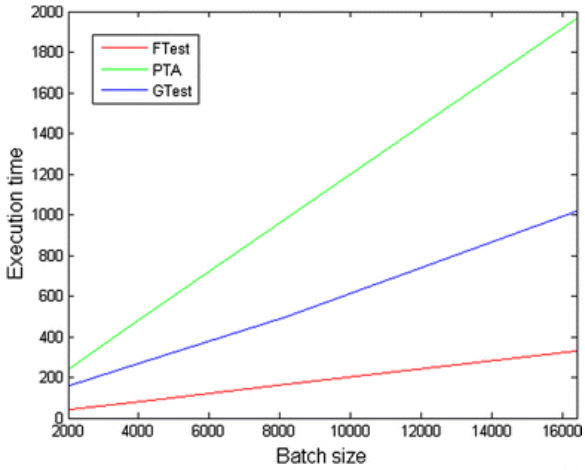
On the contrary, the data transferred in FTest one random number and *f* replies. Since each echo is in the same size (1 bits), the total size:

$$d\_\text{size}_{\text{FTest}} = (160 \times n/2 + (160 + 1) \times (n - n_{\text{rem}}) + (160 + 160) \times n_{\text{rem}})\text{bits}$$

Figure 7a, b shows the execution time of our two protocols. The figure shows that GTest performs better than PTA and FTest performs the best. We can see that FTest protocol significantly reduces the execution time. For system with $N = 2^{16}$, FTest reduces almost 800 s than PTA for the largest batch. And for system with $N = 2^{20}$, FTest reduces almost 1,700 s than PTA for the largest batch (see Table 4).



(a) Execution time of FTest, GTest, and PTA when N =2^{16}



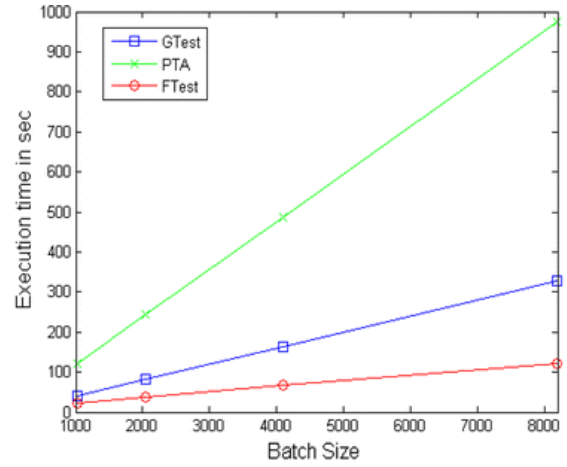
(b) Execution time of FTest, GTest, and PTA when N =2^{20}

Fig. 7 Comparison of execution time for FTest, GTest, and PTA protocol

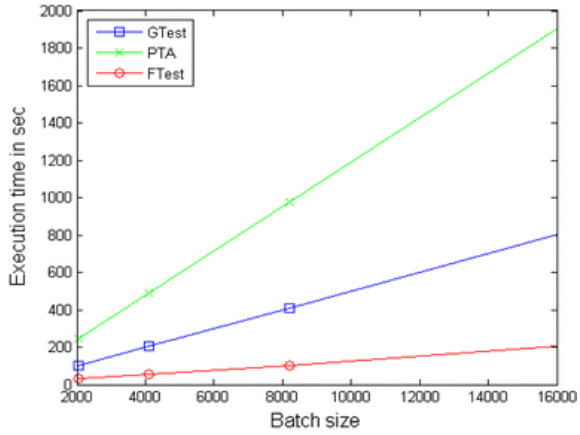
Table 4 Performance comparison table

N	**FTest savings over PTA [6]**
2^{16}	≈800 s
2^{20}	≈1,700 s

Figure 8a, b shows the execution time of our two protocols when Δ ≈ 3 %. By Δ ≈ 3 %, we mean that the protocol will consider a batch invalid if it can identify at least 3 % of the tags as counterfeit. At that point, the protocol will declare the entire batch as invalid and will stop executing further. The figure clearly shows that, with Δ ≈ 3 %, FTest is the most efficient protocol. We can see that out approaches, especially FTest protocol significantly reduces the execution time. FTest with Δ ≈ 3 % saves almost 85 s for largest batch size.



(a) Execution time of FTest, GTest, and PTA when N =2^{16} and Δ = 3%



(b) Execution time of FTest, GTest, and PTA when N =2^{20} and Δ = 3%

Fig. 8 Comparison of execution time for FTest, GTest, and PTA protocol with Δ = 3 %

Related work

In literature, extensive research works have been proposed for different types of RFID applications and to ensure different set of goals. Most of the previous works on RFID systems concentrate on collecting the ids of a large number of tags as quickly as possible. The main challenge is to resolve radio contention when the tags compete for the same low-bandwidth channel to report their ids. Other work studies the *tag-estimation problem*, which is to use statistical methods to estimate the number of tags in a large system [16]. Tan et al. [17] design the trust reader protocol (TRP) to detect the missing tags with probability when the number of missing tags exceeds a certain threshold. TRP uses probabilistic method to choose a frame size that satisfies the system requirement.

Collision is a critical problem in RFID systems when processing a batch of tags. In the literature, tag anti-collision algorithms can be categorized into Aloha-based algorithms and tree-based algorithms. Aloha-based algorithms makes only one tag respond in a slot, in the response of tags, by dividing a time into slot units. On the other hand, tree-based algorithms [18] make trees while performing the tag identification procedure using a unique id of each tag. Aloha-based protocols are known for their low complexity and computation, thus making them attractive for use in RFID networks. Examples include pure, slotted and framed slotted Aloha (FSA), and their variants [19–21]. In pure and slotted Aloha, a tag responds after a random delay, and continues doing so until it is identified. Lee et al. [7] show that the FSA reader can obtain a maximum identification throughput when the size of detecting frame equals to the number of tags and propose a dynamic FSA for RFID systems. Sheng et al. [22] study a fundamental problem of continuous scanning in RFID systems and designs algorithms based on the

information gathered in the previous scanning. Yang et al. [23] proposes a probabilistic approach, SEBA, for fast and reliable batch authentications in RFID application. However, in this protocol, when queried by a reader, tags replies with some bits of their secret ids. But the drawback of this protocol is that any adversary eavesdropping in the channel may learn complete ids of tags over time and launch several successful attacks.

Many approaches have been proposed for RFID private authentication and they can be classified into two categories, non-tree-based approaches and tree-based approaches. Non-tree-based protocols usually perform linear search to find out a tag. The search complexity is $O(N)$, where N is the number of tags. Obviously, the linear search is not efficient in large-scale RFID systems that may have millions of tags [24]. Another non-tree-based approach, Hash-lock [25] method, uses the hash value of a key to identify a tag. A variation of Hash lock needs exhaustive search through all ids to identify a tag. Molnar and Wagner proposed a tree-based approach in [3] that reduces the complexity of authentication from $O(N)$ to $O(\log N)$. This reduction is made possible by using a key tree instead of a flat key space. The level of privacy provided by the scheme is decreases quickly as more and more tags are compromised. Numbers of research have been conducted to find out a trade-off between the complexity and the level of privacy provided by the key tree-based scheme. This trade-off is identified and analyzed by Avoine et al. [26] and Buttyan et al. [27], and more recently by Nohl and Evans [28]. Avoine et al. [29] proposed a group-based private authentication scheme in that improves the trade-off between scalability and privacy by dividing the tags into a number of groups. A lightweight RFID private authentication protocol, RWP, have been proposed in [30], based on the random walk concept. The analysis results show that RWP effectively enhances the security protection for RFID private authentication, and increases the authentication efficiency from $O(\log N)$ to $O(1)$. However, this technique is suitable for tags with high computational power as the technique requires tags to perform randomized hash functions. Besides these types of deterministic approaches, some RFID application uses probabilistic methods to determine some important feature related to the system [32, 33].

Conclusions

Detecting counterfeit tags in large-scale RFID systems is a very significant but underrated research issue. Most of the existing RFID authentication methods used for counterfeit detection require a pre-identification process, and suffer from high scanning cost and communication cost. We believe that an efficient, secure and fast counterfeit detection protocol may have good impact on the deployment of many large-scale RFID systems. In this paper, we present an efficient batch authentication protocol (FTest) to detect product counterfeiting in RFID enabled systems. Our simulation results show that our FTest can perform significantly better than per-tag authentication protocols. Future research work includes the investigation of defend mechanism against concealing attack and privacy-related issues.

References

1. Pollinger ZA (2008) Counterfeit goods and their potential financing of international terrorism. Mich J Bus 1(1):85–102
2. Juels A (2005) RFID security and privacy: a research survey. Manuscript
3. Weis S, Sarma S, Rivest R, Engels D et al (2004) Security and privacy aspects of low-cost radio frequency identification systems. Lect Notes Comput Sci
4. Dimitriou T (2006) A secure and efficient RFID protocol that could make Big Brother (partially) obsolete. In: Proceedings of IEEE PerCom'06. WA, USA, pp 269–275
5. Lu L, Han J, Hu L, Liu Y, Ni LM et al (2007) Dynamic key-updating: privacy-preserving authentication for RFID systems. In: Proceedings of IEEE PerCom'07, NY, USA, pp 13–22
6. Hoque ME, Rahman F, Ahamed SI et al (2011) AnonPri: an efficient anonymous private authentication protocol. In: Proceedings of IEEE PerCom 11, WA, USA, pp 102–110

7. Lee SR, Joo SD, Lee CW et al (2005) An enhanced dynamic framed slotted aloha algorithm for RFID tag identification. In: Proceedings of IEEE MobiQuitous
8. Syamsuddin I, Dillon T, Chang E, Han S et al (2008) A survey of RFID authentication protocols based on Hash-Chain method. In: Proceedings of the third international conference on convergence and hybrid information technology
9. Lu L, Han J, Xiao R, Liu Y et al (2009) ACTION: breaking the privacy barrier for RFID systems. In: Proceedings of IEEE INFOCOM, pp 1953–1961
10. Avoine G, Oechslin P (2005) A scalable and provably secure hash based RFID protocol. In Proceedings of PerCom workshop-PerSec 05, pp 110–114
11. Hoque ME, Rahman F, Ahamed SI, Park JH et al (2009) Enhancing privacy and security of RFID system with serverless authentication and search protocols in pervasive environments. Wirel Pers Commun 55(1):65–79
12. Hoque ME, Rahman F, Ahamed SI et al (2009) Supporting recovery, privacy and security in RFID systems using a robust authentication protocol. In: Proceedings of SAC 2009, USA, pp 1062–1066
13. Ahamed SI, Rahman F, Hoque ME, Kawsar F, Nakajima T et al (2008) YA-SRAP: yet another serverless RFID authentication protocol. In: Proceedings of international conference on intelligent environment (IE 2008), USA, pp 1–8
14. EPCglobal radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz–960 MHz. (Last accessed: 20 May 2012). http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2/uhfc1g2_1_0_9-standard-20050126.pdf
15. O'Neill M (2008) Low-cost SHA-1 hash function architecture for RFID tags. In: Workshop on RFID security (RFIDSec'08)
16. Kodialam M, Nandagopal T et al (2006) Fast and reliable estimation schemes in RFID systems. In Proceedings of ACM MOBICOM, pp 322–333
17. Tan CC, Sheng B, Li Q et al (2008) How to monitor for missing RFID tags. In: Proceedings of IEEE ICDCS, pp 295–302
18. Lee DW, Bang OK, Im SY, Lee HJ et al (2008) Dual bias Q-algorithm and optimum weights for EPC class1 generation 2 protocol. In: 2008 European wireless, pp 1–5
19. Zhen B, Kobayashi M, Shimizui M et al (2005) Framed Aloha for multiple RFID objects ID. IEICE Trans Commun 991–999
20. E. Microelectronics. Supertag category protocols. Datasheet. (Last accessed 20 May, 2012). http://www.gaw.ru/doc/EM-Marin/P4022.PDF
21. Abraham C, Ahuja V, Ghosh A, Pakanati P et al (2002) Inventory management using passive RFID tags: a survey. Department of Computer Science, The University of Texas at Dallas, Richardson, TX, pp 1–16
22. Sheng B, Li Q, Mao W et al (2009) Efficient continuous scanning in RFID systems. In: Proceedings of IEEE INFOCOM, pp 1–9
23. Yang L, Han J, Qi Y, Liu Y et al (2010) Identification-free batch authentication for RFID tags. In: Proceedings of IEEE ICNP, pp 154–163
24. Roussos G, Kostakos V et al (2008) RFID in pervasive computing: state-of-the-art and outlook. J Pervasive Mob Comput 5(1):110–131
25. Molnar D, Wagner D et al (2004) Privacy and security in library RFID: issues, practices, and architectures. In: Proceedings of computer and communications security (CCS 04), USA, pp 210–219
26. Avoine G, Dysli E, Oechslin P et al (2005) Reducing time complexity in RFID systems. Sel Areas Cryptogr 3897:291–306
27. Buttyan L, Holczer T, Vajda, I Optimal key-trees for tree-based private authentication. In: Proceedings of privacy enhancing technologies workshop (PET 06), pp 332–350
28. Nohl K, Evans D et al (2006) Quantifying information leakage in tree-based hash protocols. In: Proceedings of information and communications security (ICICS 06), pp 228–237
29. Avoine G, Buttyan L, Holczer T, Vajda I et al (2007) Group-based private authentication. In: Proceedings of world of wireless, mobile and multimedia networks (WoWMoM 07), Finland, pp 1–6

30. Yao Q, Qi Y, Han J, Zhao J, Li X, Liu Y et al (2009) Randomizing RFID private authentication. In Proceedings of pervasive computing and communications workshop (PerCom Workshops 09), pp 1–10
31. Rahman F, Ahamed SI et al (2012) Looking for needles in a haystack: detecting counterfeits in large scale RFID systems using batch authentication protocol. In: Proceedings of IEEE PerCom workshop on pervasive wireless networking (PWN12)
32. Sheng B, Tan CC, Li Q, Mao W et al (2008). Finding popular categories for RFID tags. In: Proceedings of ACM MobiHoc, pp 159–168
33. Qian C, Ngan H, Liu Y et al (2008) Cardinality estimation for large-scale RFID systems. In: Proceedings of IEEE PerCom (Percom 08), pp 30–39