Looking for needles in a haystack: Detecting Counterfeits in Large Scale RFID Systems using Batch Authentication Protocol

Farzana Rahman and Sheikh Iqbal Ahamed
Department of MSCS, Marquette University, Milwaukee, WI, USA
Email: {framhan, iq}@mscs.mu.edu

Abstract—RFID is a promising technology for anti-counterfeiting since it facilitates processing of product information. In large scale RFID applications (such as supply chain, retail industry, pharmaceutical industry, etc.) tag authentication is used to detect counterfeit products. However, RFID authentication protocols are mainly per-tag based where reader needs to authenticate tags sequentially one at a time. This increases the protocol execution time due to large volume of authentication data. In this paper, we propose to detect counterfeit tags in large scale system using efficient batch authentication. 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, e.g. authentication techniques.

Keywords- RFID, Batch Authentication, Anti-Counterfeiting.

I. 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]. Many companies already use anti-counterfeiting measures like holograms to confuse counterfeiting. A drawback of existing anti-counterfeiting measures is the low achievable degree of automation when checking the originality of a product. With existing schemes, large-scale scanning is not feasible. Radio Frequency Identification, or RFID, helps to address this problem, and provides the possibility to implement secure protection mechanisms [2].

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 Farmed 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 rest of this paper is organized as follows. Section II describes our major motivation. We introduce preliminary knowledge about different authentication and anti-collision protocols in Section III. Our system model and assumptions are mentioned in section IV. In section V and section VI, we describe the sampling based GTest and FSA based FTest batch authentication protocols respectively. We discuss the security analysis of FTest protocol in section VII. We examine the performance of FTest and GTest protocol in section VIII. This section is followed by the relevant related work in Section IX. Finally, we conclude the paper with some future works in section X.

II. 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 care security, social awareness and global trading concerns.

III. BACKGROUND

Since our FTesT protocol is partially based on Framed Slotted Aloha (FSA), in this section, we discuss how and FSA and Tree based authentication protocol works.

A. Tree Based Authentication Protocol

![Figure 1. Tree based hash protocol with N=8 and α=2](image)

In tree based hash protocol [7-9], the tags are organized in a secret key tree where each tag is assigned to a leaf 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. So if the branching factor is \( \alpha \), the \( \log_{\alpha} N \) will be equal to \( L \). Figure 1 shows a balanced key tree 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

\[
h(k_{1,i}, n_r), h(k_{1,i,2}, n_r), ..., h(k_{1,i,2,...,L}, n_r),
\]

where each \( l_i \in \{1, ..., \alpha\}, 1 \leq i \leq L \) and \( h(\cdot) \) is a hash function. In this protocol, the reader’s complexity is reduced to \( O(\log_{\alpha} N) \). The major drawback of this approach is that each tag must transfer \( \log_{\alpha} N \) 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.

B. Framed Slotted Aloha

Another approach to reduce the protocol execution time is to reduce the number of collision during authentication. Aloha based protocols can significantly reduce the probability of occurrence of tag collision. In slotted Aloha [10], tags select the timeslot randomly and reply at the beginning of the timeslot to avoid overlapping of transmissions. FSA based tag identification [7] 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 tag uses a hash function \( h(x) \), which is used to choose the slot number. After receiving \( f \), each tag selects \( h(id \oplus r) \mod f \), 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. In this protocol, 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, and collision slot respectively. Slots can also be characterized as multi-bit slots and single-bit slots.

IV. SYSTEM MODEL AND PROBLEM FORMULATION

A. 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 \( \tau \) is the number of groups. So, the group size is \( n = \frac{N}{\tau} \). The number of tags in a batch, \( n_t \), 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 \( id_t \), 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 \( 1 \leq i \leq \tau \).

![Figure 2. The group organization with N=8 and \tau=4](image)

B. 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 \( id_t \) 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 and it has powerful computing capability.
C. Preliminaries and Assumptions

We assume that the reader and all the tags in the system has the knowledge of XOR operation and \(h(.)\), an irreversible one way hash function to protect the integrity of the message. The outputs of \(h(.)\) cannot be linked back to its input so that an adversary cannot link back the tag ids. In this paper, we use SHA-1 hash function that outputs 160 bits.

D. Protocol goal

The goal of a server is to accurately and efficiently determine the validity of a batch of tags. 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 (\(\Delta\)). \(\Delta\) 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.

V. 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. 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.

A. 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.

B. 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 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

\[ \delta_t = 1 - (1 - \delta)^t \]

\(\delta\) is probability that a pooled sample is positive

Here, \(\delta\) 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 \(n_x\) counterfeit tags. Therefore, \(\delta = n_x/N\)

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 \(\eta\) of the pooled samples may be invalid is given by

\[ f(\eta | n_c) = \binom{m}{\eta} \prod_{i=0}^{\eta} \left( \frac{\eta}{\eta} \right) \left( \frac{N-n_c}{N-\eta} \right) \left( \frac{n_c}{N-\eta-1} \right) \ldots (1) \]

In equation 1, when \(\eta = 0\), we derive 

\[ f(0 | n_c) = \binom{N-n_c}{n} / \binom{N}{n} \ldots (2) \]

Equation 2 refers to the hypergeometric probability since the absence of counterfeit among the \(n\) samples is equivalent to the absence of positive pooled samples. Now, 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 \(n_x\) at a time. Then the pdf of \(X\) will be:

\[ f(X | n \Delta) = \binom{n \Delta}{X} \frac{n(1-\Delta)}{N-X} \]

However, using GTest protocol, if we want to test the validity of a batch, reader needs to read a large amount of data. For example, with \(n = 100000\), \(n_x = 1000\) and \(\Delta = 0\%\) (meaning that all counterfeit needs to be detected), reader needs to read \(1000 \times 20 \times \log_2^{100000} = 3.1\) M byte of data which will take high response time. So, to decrease response time, we propose an efficient protocol in next the section.

VI. FSA Based (FTest) Batch Authentication

In this section, we propose FTest protocol that is partially dependent on FSA. We define a set of \(N\) RFID tags as \(T\).

A. FTest Protocol Design

FTest has three phases - 1) Group identification, 2) Authentication initialization and 3) Counterfeit detection. The first phase is similar to the one mentioned in GTest protocol. The other two phases are discussed next.

Authentication initialization: At this phase, 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. 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). So, the time duration of all slots in our approach 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 can be three elements in each slot of an \(RV = 0, 1, \text{ and collision}\), representing empty slot, single reply slot, and collided slot.
Algorithm 1: Algorithm executed by RFID tags

Receive \((f, r)\) from \(R\)
for each tag \(T_i\) (where \(i = 1\) to \(N\))
\[ \text{compute } SP_i = h(id_i, r) \mod f \]
end
while \(R\) broadcasts Slot Position (SP)
if \((SP = SP_i)\)
\[ \text{return } 1 \text{ in } RV[SP] \]
end

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 \( RV[SP] = \text{collision} \)
end
return \(RV\) to the server

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[i] == 0)
continue
else if (RV[i] == 1 & RV[i] == 0)
assert counterfeit detected
total_counterfeit_tag +=
else if (RV[i] == collision & RV[i] == 1)
assert counterfeit detected
else continue
end

Counterfeit detection: In this phase, 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: \(h(id || n_r)\).

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 it selves 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 responding 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.

Counterfeit detection process in FTest Protocol

Since our goal is to declare a batch invalid if the percentage of counterfeit tags exceeds counterfeit threshold \(\Delta\), 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 \(\Delta\). 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 \(n = 1000\), number of counterfeit tag detected during per tag authentication is 3. Number of counterfeit tags detected during the first 70 rounds of counterfeit detection protocol is 15. Then,

\[ \text{total_counterfeit_tag } / n = \frac{50}{1000} = 0.05 \]

If \(\Delta = 5\%\), then the counterfeit detection protocol will stop after 70th round. The complete counterfeit detection algorithm is shown in figure 5. With \(n = 100000\), \(n_r = 1000\) and \(\Delta = 0\%\), FTest reads 0.03 M byte of data per batch.

B. 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 $R_V$ and $R_V'$. There can be following types of distinguishable situations that indicate the existence of collision attack:

- When $R_V[i] = 0$ and $R_V'[i] = 1$, 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 $R_V[i] = 0$ and $R_V'[i] = \text{collision}$, there should be no genuine tags replying in this slot. But the result shows more than one tag of this batch has chosen this slot. This also ensures that there are counterfeit tags in the system.
- When $R_V'[i] = 1$ and $R_V[i] = \text{collision}$, 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.

VII. SECURITY ANALYSIS OF FTest

A. 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. We assume $\hat{A}$ is an adversary who can eavesdrop in between the channel and can use the learned information to create counterfeit tags and install them in the system. Each counterfeit tag is denoted as $\hat{T}$. The attacker may try to track genuine tags of the system. We also assume that genuine tags and reader cannot be compromised by the attacker. In our system, the following oracle-like construction exists:

$O_{\text{Eavesdrop}}(R, T, t)$: The adversary eavesdrops to listen to the communication between R and one of its tag T.

$O_{\text{Impersonate}}(R, T, M, t)$: The adversary impersonates a reader R by sending a message M to the tag T.

$O_{\text{Impersonate}_{\text{wT}}}(R, T, M, t)$: The counterfeit tag T impersonates a genuine tag in a protocol session at time t and sends a message M to the reader R.

$O_{\text{Query}}(T, t)$: The adversary queries a tag T to learn information during a communication session at time t.

$O_{\text{Receive}}(U, M, t)$: The adversary receives a message M from an entity U (e.g., either T or R) at time t.

B. Security Analysis

Each attack and defend have three phases: 1) Learning, 2) Attacking, and 3) Defend. Using the above mentioned oracles, we can also prove that FTest is secure against eavesdropping, replay, tracking, and collision attack. But, we do not discuss those in this paper due to space limitation.

VIII. SIMULATION

We evaluate the efficiency of GTest and FTest Protocol 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, and 11-13]. 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 $N = 2^{16}$, $\tau = 8, 16, 32, 64$ and $N = 2^{20}$, $\tau = 512, 256, 128, 64$. We deliberately introduce 1% to 4% randomly generated counterfeits into the dataset. We have run the simulation for 100 times and reported the average.

![Graph 1](image1)

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

![Graph 2](image2)

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

**Figure 6. Execution time of FTest, GTest and PTA protocol**

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 when we need to identify all the counterfeit tags in each batch. Since every bit almost consumes the same transmission time which equals 25µs [14] on average, we measure the execution time by multiplying the size of transmitted data (in bits) with 25µs. 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 $160 \times n/2$ (since tags will reply with probably 0.5). Now to verify a batch with n tags, suppose that $n_T$ tags are sampled and the length of random numbers equals 160 bits. The size of authentication phase will equal $(160 + 160) \times n_T$ bits, since reader will challenge with a random number (160 bits) and tags will reply with their hashed response (160 bits). So, the total data size of GTest protocol will be:
\[ d_{\text{size}_{\text{GTest}}} = \left( 160 \times \frac{n}{2} + (160 + 160) \times n_r \right) \text{bits} \]

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 (1bits), the total size:

\[ d_{\text{size}_{\text{FTest}}} = \left( 160 \times \frac{n}{2} + (160 + 1) \times (n - n_{\text{rem}}) + (160 + 160) \times n_{\text{rem}} \right) \text{bits} \]

Figure 6(a) and 6(b) shows the execution time of our two protocols. The figure shows that GTTest performs better than PTA and FTest performs the best. We can see that FTest protocol significantly reduces the execution time. For system with \( N = 2^{10} \), FTest reduces almost 800 sec than PTA for the largest batch. And for system with \( N = 2^{20} \), FTest reduces almost 1700 sec than PTA for the largest batch.

**IX. RELATED WORK**

Most previous works on RFID systems concentrate on collecting the ids of a large number of tags. Other work studies the tag-estimation problem, which is to use statistical methods to estimate the number of tags in a large system [16, 22]. Tan, Sheng and Li [17] design the Trust Reader Protocol (TRP) to detect the missing tags with probability when the number of missing tags exceeds a certain threshold. In literature, tag anti-collision algorithms can be categorized into Aloha based algorithms and tree based algorithms. 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 suitable for RFID systems. Examples include Pure, Slotted and Framed Slotted Aloha (FSA), and their variants [10, 19, 20, and 21]. Yang el al. [23] proposes a probabilistic approach,SEBA, for fast and reliable batch authentications in RFID application. The drawback of this protocol is that any adversary can learn complete ids of tags over time by listening in the channel and can launch several successful attacks. Many approaches have been proposed for RFID private authentication [6, 11-13, 24] and they can be classified into two categories, non-tree-based and tree-based approaches. Non-tree-based protocols usually perform linear search and the complexity is \( O(N) \) [25]. Molnar and Wagner proposed a tree based approach in [26] that reduces the complexity from \( O(N) \) to \( O(\log N) \).

**X. CONCLUSION**

Most of the existing RFID counterfeit detection technique require a pre-identification process, and suffer from high execution time. In this paper, we present an efficient batch authentication protocol (FTest) to detect product counterfeiting in RFID enabled large systems. Our simulation results show that FTest can perform significantly better than per tag authentication protocols. Future research include the investigation of defend mechanism against concealing attack and privacy issues.