2015 IEEE 23rd International Conference on Network Protocols

ETAP: Enable Lightweight Anonymous RFID Authentication with $O(1)$ Overhead

Min Chen Shigang Chen
Department of Computer & Information Science & Engineering
University of Florida, Gainesville, FL 32611, USA
Email:{min, sgchen}@cise.ufl.edu

***Abstract*—Radio frequency identification (RFID) technologies are making their way into retail products, library books, debit cards, passports, driver licenses, car plates, medical devices, etc. The widespread use of tags in traditional ways of deployment raises a privacy concern: They make their carriers trackable. To protect the privacy of the tag carriers, we need to invent new mechanisms that keep the usefulness of tags while doing so anonymously. Many tag applications such as toll payment require authentication. This paper studies the problem of anonymous authentication. Since low-cost tags have extremely limited hardware resource, we propose an asymmetric design principle that pushes most complexity to more powerful RFID readers. Thus, we develop a lightweight technique that generates dynamic tokens for anonymous authentication. Instead of implementing complicated and hardware-intensive cryptographic hash functions, our authentication protocol only requires tags to perform several simple and hardware-efficient operations such as bitwise XOR, one-bit left circular shift, and bit flip. The theoretic analysis and randomness tests demonstrate that our protocol can ensure the privacy of the tags. Moreover, our protocol reduces the communication overhead and online computation overhead to $O(1)$ per authentication for both tags and readers, which compares favorably with the prior art.**

I. Introduction

Radio frequency identification (RFID) technologies integrate simple communication, storage and computation components in attachable tags that can communicate with readers wirelessly over a distance [1]–[3]. Each tag uniquely identifies its carrier, which can be a product in a warehouse, a merchandize in a retail store, an animal in a zoo, or a piece of medical equipment in a hospital. Active research in recent years has been continuously expanding the RFID application scope, [4]–[9], and practical RFID systems are applied to inventory and logistics management, object tracking, access control, automatic toll payment, theft prevention, localization, intelligent transportation systems, etc. The market size of RFID has reached $8.89 billion in 2014, and is projected to rise to $27.31 billion after a decade according to a market research conducted by IDTechEx [10].

The proliferation of tags in their traditional ways of deployment is introducing a hidden problem: They make their carriers trackable. Should future tags penetrate into everyday products and be carried around (oftentimes unknowingly), people's privacy would become a serious concern. A typical tag will automatically transmit its ID in response to the query from a nearby reader. If we carry tags in our pockets or by our cars, these tags will give off their IDs to any readers that query them, allowing others to track us. As an example, for a person who carries a tag in her purse (a tagged card or a smart phone that implements tag function), she may be unknowingly tracked by retailers equipped with readers in the stores to learn when she visits which product section for how long. For a person whose car carries a tag (automatic toll payment [11] or tagged plate [12]), he may be unknowingly tracked over years by toll booths or others who install readers at locations of interest to learn when and where he has been. To protect the privacy of tag carriers, we need to invent ways of keeping the usefulness of tags while doing so anonymously.

Many RFID applications such as toll payment require authentication. A reader will accept a tag's information only after authenticating the tag and vice versa. What is the challenge to make authentication anonymous? Let us answer this question through an example: Consider an RFID tag for toll payment such as SunPass [11]. Suppose the reader has access to a database of all secret keys that are pre-installed in the toll-payment tags. When a vehicle approaches, the reader has to know which key it should use to perform authentication. In a typical authentication protocol, the tag transmits a key identifier (i.e., user ID) to the reader, which allows the reader to identify the right key. However, the key identifier, unique to each tag, can be used to identify the carrier. Unauthorized readers from an adversary can initiate the authentication process at any chosen locations and obtain the key identifiers of passing vehicles, which allows them to track the whereabout of the vehicles. Therefore, anonymous authentication should prohibit the transmission any identifying information, such as tag ID, key identifier or any fixed number that may be used for identification purpose. As a result, there comes the challenge that how can a legitimate reader efficiently identify the right key for authentication without any identifying information of the tag?

The importance and challenge of anonymous authentication attract much attention from the RFID research community. Many anonymous authentication protocols have been proposed. However, all prior work has some potential problems, either incurring high computation or communication overhead, or having security or functional concern (will be explained shortly). Moreover, most prior work, if not all, employs cryptographic hash functions, which requires considerable hardware [13], to randomize authentication data in order to make the tags untrackable. The high hardware

1092-1648/15 $31.00 © 2015 IEEE
DOI 10.1109/ICNP.2015.21

requirement makes them not suited for low-cost tags with limited hardware resource. Hence, designing anonymous authentication protocols for low-cost tag remains an open and challenging problem [14].

Our contribution: In this paper, we make a fundamental shift from the traditional design paradigm for anonymous RFID authentication. First, we release the resource-constrained RFID tags from implementing any complicated functions (e.g., cryptographic hashes). Since the readers are not needed in a large quantity as tags do, they can have much more hardware resource. Given the significant capability disparity between the readers and tags, we propose an asymmetry design principle which pushes most complexity to the readers while leaving the tags as simple as possible. More specifically, our protocol only requires the tags to perform a few hardware-efficient operations such as bitwise XOR, one-bit left circular shift, and bit flip, while all other complicated work is done by the readers or servers. Second, we develop a novel technique to generate random tokens on demand for anonymous authentication. Our protocol only requires $O(1)$ communication overhead and online computation overhead per authentication for both readers and tags, which is a significant improvement over the prior art. Hence, our protocol is scalable to large RFID systems. Finally, extensive theoretic analysis, security analysis, simulations and statistical randomness tests are provided to verify the effectiveness of our protocol.

II. Prior Art

Prior work on anonymous authentication can be generally classified to two categories: non-tree-based and tree-based.

A. Non-Tree based Protocols

The Hash-lock [15] takes advantage of random hash values for anonymous authentication. After receiving an authentication request from a reader, a tag sends back $(r, id \oplus f_k(r))$, where r is a random number, id is the tag's ID, k is a pre-shared secret between the tag and the reader, and $\{f_n\}_{n\in\mathbb{N}}$ is a pseudo-random number function ensemble. The reader exhaustively searches its database for a tag whose ID and key can produce a match with the received data. The hash-lock protocol has a serious efficiency problem that the reader needs to perform $O(n)$ hash computations on line per authentication, where n is the number of tags in the system. Some variants [16]–[19] of hash-lock scheme try to improve the search efficiency, but they have issues. The OKS protocol [16] uses hash-chain for anonymous authentication. The OSK/AO protocol [17], [18] leverages the time-memory tradeoff to reduce the search complexity to $O(n^{\frac{2}{3}})$ (still too large) at the cost of $O(n^{\frac{2}{3}})$ units of memory. However, both OKS and OSK/AO cannot guarantee anonymity under denial-of-service (DoS) attack [20]. The YA-TRAP protocol [19] makes use of monotonically increasing timestamps to achieve anonymous authentication. YA-TRAP is also susceptible to DoS attack, and a tag can only be authenticated once in each time unit. The DoS attack in OSK/AO and YA-TRAP is in nature a desynchronization attack, which tricks a tag into updating its keys unnecessarily and makes it fail to be authenticated by an authorized reader later.

The LAST protocol was proposed based on a weak privacy model [21]. Key identifiers are employed to facilitate the reader to identify the tags quickly. After each authentication, the reader uploads a new $\langle identifier, key\rangle$ pair to the tag. LAST only requires the reader and tag to compute $O(1)$ hashes per authentication, but the overhead for the reader to search a given key identifier is not considered. Moreover, since the key identifier is only updated after a successful authentication, the tag keeps sending the same key identifier between two consecutive successful authentications. Therefore, LAST is not anonymous in the strict sense. In addition, the process of uploading a new $\langle identifier, key\rangle$ pair to the tag after each authentication incurs extra communication overhead.

B. Tree based Protocols

Tree-based protocols organize the shared keys in a balanced tree to reduce the complexity of identifying a tag to $O(\log n)$. However, the tree-based protocols generally require each tag to store $O(\log n)$ keys, which is $O(1)$ for non-tree based protocols.

In Dimitriou's protocol [22], the non-leaf nodes of the tree store auxiliary keys that can be used to infer the path leading to leaf nodes that store the authentication keys. For each authentication, the computation overhead for both the reader and the tag is $O(\log n)$, and the tag needs to transit $O(\log n)$ hash values. This protocol is is vulnerable to the compromising attack since different tags may share auxiliary keys [23], [24].

The ECNP protocol [25] leverages a cryptographic encoding technique to compress the authentication data transmitted by tags. ECNP can reduce the computation overhead of the reader and the transmission overhead of the tag by multifold compared with Dimitriou's protocol [22], but they remain $O(\log n)$ due to the use of tree structure. Moreover, ECNP is not resistant against the compromising attack since the children of one node in the tree share the same group keys.

The ACTION protocol [24] was designed to be resistant against the compromising attack. It adopts a sparse tree architecture to make the keys of each tag independent from one another. In ACTION, each tag is randomly assigned with a path key, which is further segmented into *link indices* to guide the reader to walk down the tree towards the leaf node that carries the secret key k of the tag. For each authentication, a tag needs to compute and transmit $O(\log n)$ hashes and the reader needs to perform $O(\log n)$ hashes to locate the shared key. The key problem of ACTION is that the size of link indices is too small after segmentation (e.g., 4 bits), rendering them easy to guess.

III. System Model and Security Model

A. System Model

Consider a hierarchical distributed RFID system as shown in Fig. 1. Each tag is pre-installed with some keys for authentication. The readers are deployed at chosen locations, responsible for authenticating tags entering their coverage

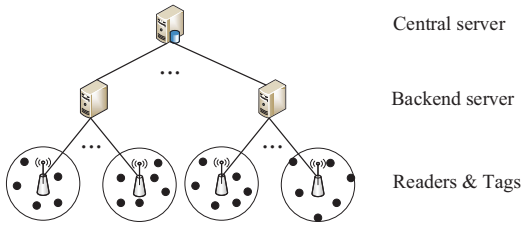

Fig. 1: A hierarchical distributed RFID system.

areas. In addition, the readers at each location are connected to a backend server, serving as a supplement to provide more storage and computation resources. All backend servers are further connected to the central server, where every tag' keys are stored. Any authorized backend server can fetch the tags' keys from the central server. Since the keys of each tag are only stored at the central server, they are synchronized from the view of different backend servers. Moreover, the high-speed links connecting the central server, backend servers and readers make the latency of transmitting small authentication data negligible. Therefore, a reader, its connected backend server, and the central server can be thought as single entity, and will be used interchangeably.

In this paper, we concentrate on low-cost RFID tags, particularly passive backscatter tags that are ubiquitously used nowadays. The simplicity of these tags contributes to their low prices, which in turn restricts their computation, communication, and storage capabilities. In contrast, the readers, which are not needed in a large quantity as tags do, can have much richer resource. Moreover, the backend server can provide the readers with extra resource when necessary. The communication between a reader and a tag works in the request-and-response mode. The reader initiates the communication by sending a request. Upon receiving the request, the tag makes an appropriate transmission in response. We divide the transmissions between the readers and tags into two types: (1) *Invariant transmissions* contain the content that is invariant between any tag and any reader, such as the beacon transmission from a reader which informs the incoming tag of what to do next. (2) *Variant transmissions* contain the content that may vary for different tags or the same tag at different times, such as the exchanged data for anonymous authentication.

B. Security Model

Threat model: An adversary may eavesdrop on any wireless transmissions made between the tags and the readers. In addition, the adversary may plant unauthorized readers at chosen locations, which communicate with passing tags and try to identify the tag carriers. However, such unauthorized readers have no access to the backend servers or the central server since the servers will authenticate the readers before granting access permissions. In the sequel, a reader without further notation means an authorized one by default. Moreover, we assume that the adversary may compromise some tags and obtain their keys, but it cannot compromise any authorized readers.

Anonymous model: The anonymous model requires that all variant transmissions must be *indistinguishable* by the adversary, meaning that (1) any variant transmission in the protocol should not carry a fixed value that is unchanged across multiple authentications, and (2) the transmission content should appear totally random and unrelated across different authentications to any eavesdropper that captures the transmissions. Therefore, no adversary will have a non-negligible advantage in successfully guessing the next variant transmission of a tag based on the previous transmissions [20].

Tag	Token Array	Token Index
t_1	$[tk_1^1, tk_1^2, \ldots, tk_1^m]$	pt_1
t_2	$[tk_2^1, tk_2^2, \ldots, tk_2^m]$	pt_2
$\vdots$	$\vdots$	$\vdots$
t_n	$[tk_n^1, tk_n^2, \ldots, tk_n^m]$	pt_n

TABLE I: Key table for the preliminary design.

IV. A Strawman Solution

Before moving to our main contributions, we propose a strawman solution for lightweight anonymous authentication.

A. Motivation

Most prior wok, if not all, employs cryptographic hash functions to randomize authentication data for the purpose of keeping anonymity. Implementing a typical cryptographic hash function such as MD4, MD5, and SHA-1 requires at least 7K logic gates [13]. However, widely-used passive tags only have 7K-15K logic gates, of which 2K-5K are reserved for security purposes [26]. The hardware constraint necessitates a new design paradigm for lightweight anonymous authentication protocols that are more supportive for low-cost tags. The commercial success of RFID tags lies with their simplicity. Although there is no specification on how simple these tags should be, it is safe to say that we will always prefer a solution that achieves the comparable goal with less hardware requirement. On the other hand, the significant disparity between the readers and tags points out an *asymmetry design principle* that we will follow: push most complexity to the readers while leaving the tags as simple as possible.

B. A Strawman Solution

Consider an RFID system with n tags $t_1, t_2, \ldots, t_n$, each pre-installed with an array of m unique random tokens, $[tk_i^1, tk_i^2, \ldots, tk_i^m]$ $(1 \leq i \leq n)$. Tag t_i also has a token index pt_i (initialized to 1) pointing to its first unused token. The tokens and token index of each tag are also stored in the database of the central server, as illustrated in Table I.

In the sequel, we consider the authentication process between an arbitrary reader R and an arbitrary tag t having a token array $[tk^1, tk^2, \ldots, tk^m]$ and a token index pt. To authenticate t, R sends a request to t. Upon receiving the request, t sends its first unused token tk^{pt} to R. After

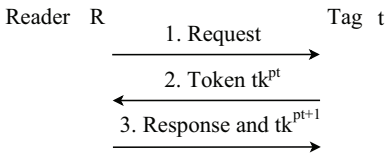

Fig. 2: Three steps of token-based mutual authentication.

that, t increases pt by 1 to guarantee that the same token will not be used twice. Otherwise, t will always send the same token when an unauthorized reader requests a token, which breaks the anonymity. After receiving the token, R has to search the token in the database since it does not know the tag's identity. Starting from $i = 1$, R checks if $tk_i^{pt_i} = tk^{pt}$ one by one. If there exists an $i \in [1, n]$ such that $tk_i^{pt_i} = tk^{pt}$, t is successfully authenticated; otherwise, t fails the authentication. In the former case, R sends back the token $tk_i^{pt_i+1}$ to t to authenticate itself, and sets $pt_i = pt_i + 2$. Tag t compares the received token with tk^{pt} to authenticate R, and increases pt by 1 again. Fig. 2 shows the three steps of the mutual authentication.

In this approach, the online computation overhead of the tag is low — only one comparison (requires far less hardware than implementing a cryptographic hash) per authentication. The online computation complexity of the reader is $O(n)$ since at most n comparisons (though one comparison is much cheaper than computing one hash value) are needed for searching the received token. The communication overhead for both the reader and the tag is $O(1)$.

To avoid the leakage of the tag's identity, the tokens used for authentication should look random. In addition, each token can be used only once. Hence, the tag must be replenished with new tokens after $\frac{m}{2}$ mutual authentications, e.g., purchasing new tokens from an authorized branch. Therefore, the tag should store as many tokens as possible to reduce the inconvenience caused by token replenishment. A low-cost tag, however, only has a tiny memory. For example, a passive UHF tag generally has a 512-bit user memory for storing user-specific data (tokens in our case). Some high-end tags with large memory [27], [28] are prohibitively expensive to be applied in large quantities. As an example, *x Sky-ID* tags [27] with an 8KB user memory cost $25 each. We will introduce the security issues of this design in the next section.

V. Dynamic Token based Authentication Protocol

In this section, we propose our first dynamic Token based Authentication Protocol (TAP). TAP can produce tokens for anonymous authentication on demand, and therefore does not require the tags to pre-install many tokens. However, we will show shortly that TAP still has some problems, which will be solved by our final design in the next section.

A. Motivation

Given the memory constraint, each tag can only store a few tokens. Frequent token replenishment brings about unacceptable inconvenience in practice. Hence, we want to invent a way to enable dynamic token generation from the few pre-installed tokens. In addition, the time for the reader to search a particular token is $O(n)$ in the preliminary design. We desire to reduce this overhead to $O(1)$. More importantly, we hope all advantages of the preliminary design, including no requirement of cryptographic hash functions, low computation overhead for the tag, and low communication overhead for both the reader and the tag, can be retained in our new design.

B. Overview

Let an arbitrary tag t in the system be pre-installed with u *base tokens*, denoted by $[bt^1, bt^2, \ldots, bt^u]$, each being a-bit long. These base tokens can be used to derive dynamic tokens for authentication. In addition, we introduce another type of auxiliary keys called *base indicators* to generate *indicators* that support the derivation of dynamic tokens. Suppose t stores v base indicators denoted by $[bi^1, bi^2, \ldots, bi^v]$, each being b-bit long. Let tk represent the current a-bit token, and ic be the current b-bit indicator. All the base tokens, base indicators, token and indictor are also stored at the central server. Our idea is to let the reader and the tag independently generate the same random tokens by following the instruction encoded in the indicator. TAP consists of three phases: *initialization phase*, *authentication phase*, and *updating phase*, which will be elaborated one by one.

C. Initialization Phase

The central server stores all tags' keys in a *key table*, denoted by KT. As shown in Table II, each entry is indexed by the *tag index*, supporting random access in $O(1)$ time. With the tag index idx, the keys of t can be found at $KT[idx]$.

When t joins the system, the reader randomly generates an array of u base tokens $[bt^1, bt^2, \ldots, bt^u]$, an array of v base indicators $[bi^1, bi^2, \ldots, bi^v]$, a token tk and an indicator ic for t. After that, the reader requests the central server to store those keys of t in the database. The central server inserts the keys to the first empty entry in KT. The search process for an empty entry can be sped up by maintaining a small table recording all empty entries in KT (e.g., due to tags' departure). If KT is fully occupied, the central server doubles its size to accommodate more tags.

To identify a tag based on its token in $O(1)$ time, the central server maintains a hash table HT, mapping the token of each tag to its tag index. Let HT consist of l slots. At first, every slot in HT is initialized to zero. After t joins the system, the reader computes the hash value $h(tk)$, where the hash function $h(\cdot)$ yields random values in $[1, l]$, and then puts the tag index idx of t in the $h(tk)$th slot of the hash table, i.e., $HT[h(tk)] = idx$ (the potential problem of hash collisions will be addressed shortly). Fig. 3 presents an illustration of the hash table built for the tokens of five tags.

D. Authentication Phase

The authentication process of TAP is similar to that of the preliminary design as shown in Fig. 2. One difference is that the reader can quickly identify the tag from its token using

Tag Index	Tag	Base Token Array	Token	Base Indicator Array	Indicator
1	t_1	$[bt_1^1, \ldots, bt_1^u]$	tk_1	$[bi_1^1, \ldots, bi_1^v]$	ic_1
2	t_2	$[bt_2^1, \ldots, bt_2^u]$	tk_2	$[bi_2^1, \ldots, bi_2^v]$	ic_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	t_n	$[bt_n^1, \ldots, bt_n^u]$	tk_n	$[bi_n^1, \ldots, bi_n^v]$	ic_n

TABLE II: Key table stored by the central server for TAP.

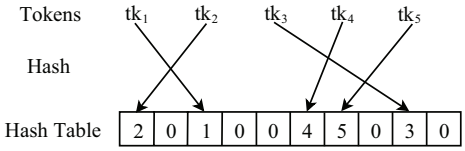

Fig. 3: A hash table used by TAP. The tokens of the five tags t_1, t_2, t_3, t_4, t_5 are tk_1, tk_2, tk_3, tk_4, tk_5, respectively. Each token is randomly mapped to a slot in the hash table, where the corresponding tag index is stored.

the hash table. After receiving a token tk from tag t, the reader first calculates $h(tk)$, and then accesses $HT[h(tk)]$ to retrieve the tag index of t, which is idx. If the reader finds $idx = 0$, it asserts the tag is fake and informs the tag of authentication failure. Otherwise, the reader refers to $KT[idx]$ in the key table to fetch the token, and compares it with the received token tk. Only if the two tokens are identical will the tag pass the authentication. In case that t is successfully authenticated, the reader will generate and transmit a new token to authenticate itself. The generation of tokens with good randomness requires the reader (more exactly, the central server) and the tag to update their shared keys synchronously.

E. Updating Phase

To guarantee anonymity, the tokens exchanged between the reader and the tag should have good randomness. Therefore, the reader (central server) and the tag need to synchronously update their shared keys after the current token is used. We stress that the tag will update its keys once it uses its current token. Therefore, the same token will never be used in two consecutive authentications, which fundamentally differs from LAST [21] where the tag only updates its key identifier after a successful authentication (which breaks the anonymity).

The tag t relies on its current indicator ic to update its keys. Fig. 4 shows the structure of an indicator, which includes two parts: The low-order $(b-2)$ bits form a *selector*, indicating which base tokens/base indicators should be used to derive the new token/indicator, while the high-order two bits encode the *update pattern*. When the updating phase begins, t calculates a

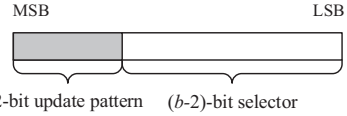

Fig. 4: The Structure of a b-bit indicator.

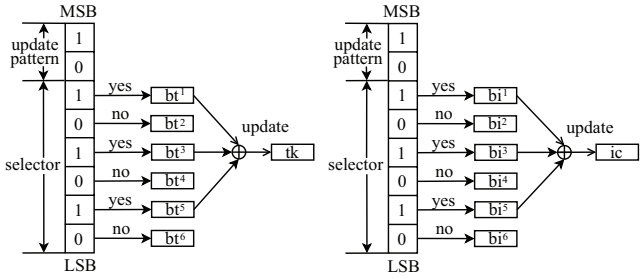

Fig. 5: *Left plot*: Generation of a new token using the base tokens and the selector. *Right plot*: Generation of a new indicator using the base indicators and the selector.

new token from the base tokens according to the selector. Each of the low-order u bits ($u \leq b-2$) in the selector encodes a choice of the corresponding base token: '0' means *not selected*, while '1' means *selected*. For all selected base tokens, they are XORed to compute the new token. Therefore,

$$tk = \bigoplus_{j=1}^{u} ic[j] bk^j, \tag{1}$$

where $ic[j]$ is the jth bit in ic (assume one-based indexes are used) and $\oplus$ is the XOR operator. The left plot in Fig. 5 gives an example of token update, where bt^1, bt^3, and bt^5 among the six base tokens happen to be selected. Similarly, t derives a new indicator from the base indicators as follows:

$$ic = \bigoplus_{j=1}^{v} ic[j] bi^j. \tag{2}$$

At the server's side, the same new token and new indicator can be generated because it shares the same keys with the tag. In addition, the server also needs to update the hash table. First, the server sets $HT[h(tk)]$ (the old token) to 0, and after generating the new token, it sets $HT[h(tk)] = idx$.

After updating the token and the indicator, the central server and the tag need to further update the selected base tokens and base indicators. The update process for any selected base token or base indicator includes two steps: A one-bit left circular shift, and bit flip by following the particular 2-bit update pattern:

1) Pattern $(00)_2$: no flip is performed;
2) Pattern $(01)_2$: flip the jth bit if $j \equiv 0 \pmod 3$;
3) pattern $(10)_2$: flip the jth bit if $j \equiv 1 \pmod 3$;
4) Pattern $(11)_2$: flip the jth bit if $j \equiv 2 \pmod 3$.

Obviously, the ith and jth bits can be flipped together if and only if $i \equiv j \pmod 3$. This rationale of the updating scheme is that if the parameters a and b are set properly, any two bits in a base token or a base indicator have a chance to not be flipped together, thereby reducing their mutual dependence. We will provide the formal proof shortly. We emphasize that all keys are only stored at the central server rather than every single reader. Hence, the update process of a tag's keys triggered by one reader is transparent to other readers (a tag carrier can only appear at one location at a time.).

F. Randomness Analysis

As required by our anonymous model, the tokens generated by TAP should be random and unpredictable. Randomness is a probabilistic property that should be described in terms of probability. We first prove the following theorem.

Theorem 1. If $a \geq 2$ and $a \not\equiv 0 \pmod 3$, there must exist a positive integer w, where $1 \leq w \leq a$, such that any two different bits in one base token will move to positions that cannot be flipped together after the base token is updated w times.

Proof: We track two arbitrary bits in the base token bt^j, denoted by random variables X and $Y \in \{0, 1\}$. Suppose X and Y are initially located at the pth bit and qth bit of bt^j ($1 \leq p < q \leq a$), respectively, and w updates are performed ($1 \leq w \leq a$). Two possible cases need to be considered according to their initial positions:

Case 1: $q - p \not\equiv 0 \pmod 3$ and $a + p - q \not\equiv 0 \pmod 3$. First, if $q + w \leq a$, X and Y have moved to $bt^j[p + w]$ and $bt^j[q+w]$, respectively. Since $(q+w)-(p+w) \not\equiv 0 \pmod 3$, they cannot be flipped together. Second, if $p+w \leq a < q+w$, then X moves to $bt^j[p + w]$ and Y moves to $bt^j[q + w - a]$. Because $(p+w)-(q+w-a) \not\equiv 0 \pmod 3$, they still cannot be flipped together. Finally, if $p + w > a$, X and Y are now at $bt^j[p + w - a]$ and $bt^j[q + w - a]$, respectively. Similarly, since $(q + w - a) - (p + w - a) \not\equiv 0 \pmod 3$, they cannot be flipped together. Hence, X and Y will never be flipped together under such conditions.

Case 2: $q-p \equiv 0 \pmod 3$ or $a+p-q \equiv 0 \pmod 3$ (Note that by no means will $q-p \equiv a+p-q \equiv 0 \pmod 3$ because $a \not\equiv 0 \pmod 3$). If $q - p \equiv 0 \pmod 3$ and $a - q < w \leq a-p$, X moves to $bt^j[p+w]$ and Y moves to $bt^j[q+w-a]$. Because $(p + u) - (q + u - a) \not\equiv 0 \pmod 3$, they move to positions that cannot flipped together. On the contrary, if $a+p-q \equiv 0 \pmod 3$, X and Y will not be flipped together at the beginning, but it becomes possible after w updates as long as $a - q < w \leq a - p$.

Hence, X and Y have a chance to not be flipped together within a updates regardless of their initial positions. ■

Before investigating the randomness of the derived tokens, we first study the randomness of an arbitrary base token during its updates. We have the following lemma:

Lemma 1. *If the update pattern in the indicator is random, an arbitrary bit in a base token becomes 0 or 1 with equal probability using our update scheme.*

Proof: Let us track one arbitrary bit in bt^j, denoted by a random variable $X \in \{0, 1\}$. Suppose X is currently located at position $bt^j[i]$, where $1 \leq i \leq a$. When bt^j is updated, X is left shifted and then flipped with a probability of 0.25 if the update pattern is random. Therefore, the transition matrix for X during each update is $P_1 = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{3}{4} \end{pmatrix}$. Using singular value decomposition (SVD) [29], $P_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix}$. Hence, the transition matrix for X after w updates is ${P_1}^w = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{pmatrix}^w \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} + \frac{1}{2}^{w+1} & \frac{1}{2} - \frac{1}{2}^{w+1} \\ \frac{1}{2} - \frac{1}{2}^{w+1} & \frac{1}{2} + \frac{1}{2}^{w+1} \end{pmatrix}$, which converges to $\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$. Therefore, X becomes 0 or 1 with equal probability. ■

Now let us further investigate two arbitrary bits in a base token, and we have the following lemma:

Lemma 2. *If the update pattern in the indicator is random, two arbitrary bits in a base token are independent under our update scheme.*

Proof: Consider two arbitrary bits, denoted by random variables X and Y, in base token bt^j. Suppose X and Y are initially located at the pth bit and qth bit of bt^j ($1 \leq p < q \leq a$), respectively. The transition matrices when X and Y cannot be flipped together and can be flipped together are

$$P_2 = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & \frac{1}{2} & 0 & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{pmatrix}, \text{ and } P_3 = \begin{pmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{pmatrix},$$

respectively. Assume that among the w ($w \geq a$) updates, X and Y cannot be flipped together for β times, while can be flipped together for γ times. We know from Theorem 1 that $\beta \geq 1$, so we have $\beta \geq 1$, $\gamma \geq 0$, and $\beta + \gamma = w$.

Case 1: $\gamma > 0$. We have

$${P_2}^\beta \times {P_3}^\gamma = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

for any combinations of β and γ.

Case 2: $\gamma = 0$. Hence, ${P_2}^\beta \times {P_3}^\gamma = {P_2}^w$. Using SVD, we can calculate

$${P_2}^w = \begin{pmatrix} \frac{1}{4} + \frac{1}{2}^{w+1} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} - \frac{1}{2}^{w+1} \\ \frac{1}{4} & \frac{1}{4} + \frac{1}{2}^{w+1} & \frac{1}{4} - \frac{1}{2}^{w+1} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} - \frac{1}{2}^{w+1} & \frac{1}{4} + \frac{1}{2}^{w+1} & \frac{1}{4} \\ \frac{1}{4} - \frac{1}{2}^{w+1} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} + \frac{1}{2}^{w+1} \end{pmatrix},$$

each entry converging to $\frac{1}{4}$. Therefore, two arbitrary bits in a base token are pairwise independent. ■

With the two lemmas above, we can prove the following theorem regarding the randomness of the derived tokens.

Theorem 2. *If the indicator is random, any bit in the derived token has an equal probability to be 1 or 0, and two arbitrary bits in the derived token are independent using our update scheme.*

Proof: Consider the ith bit of the derived token tk, denoted by $tk[i]$ ($1 \leq i \leq a$). We know $tk[i] =$

$\bigoplus_{j=1}^{u} ic[j]bt^j[i]$. Let N_0 be the random variable of the number of base tokens whose ith bit is 0, and N_1 be the random variable of the number of base tokens whose ith bit is 1, subjecting to $N_0 \geq 0$, $N_1 \geq 0$ and $N_0 + N_1 = u$. According to Lemma 1 and the independence of different base tokens, N_0 follow the binomial distribution $B(u, 0.5)$, and $P(N_0 = x) = \binom{u}{x} \times \left(\frac{1}{2}\right)^u$, where $0 \leq x \leq u$. To calculate $tk[i]$, we need to consider two possible cases:

Case 1: $N_0 = u$, namely, there is no 1 in those u bits. In this case, $tk[i]$ must be 0.

Case 2: $0 \leq N_0 < u$. In this case, $tk[i]$ can be 0 or 1. If $tk[i] = 0$, it implies that an even number of base tokens whose ith bit is 1 are chosen, and the conditional probability is:

$$\begin{aligned} P(tk[i] = 0 \,|\, 0 \leq N_0 < u) =& \frac{2^{N_0} \times \sum_{x=0}^{\lceil \frac{N_1}{2} \rceil} \binom{N_1}{2x} - 1}{2^u - 1} \\ =& \frac{2^{u-1} - 1}{2^u - 1}. \end{aligned} \tag{3}$$

Hence, the probability for $tk[i] = 0$ is:

$$\begin{aligned} P(tk[i] = 0) =& P(N_0 = u) \times P(tk[i] = 0 \,|\, N_0 = u) \\ &+ P(0 \leq N_0 < u) \times P(tk[i] = 0 \,|\, 0 \leq N_0 < u) \\ =& \frac{1}{2^u} \times 1 + (1 - \frac{1}{2^u}) \times \frac{2^{u-1} - 1}{2^u - 1} = \frac{1}{2}. \end{aligned} \tag{4}$$

Therefore, $P(tk[i] = 1) = P(tk[i] = 0) = \frac{1}{2}$. Moreover, because $tk[i]$ is determined only by the ith bits of the base tokens, and two arbitrary bits in a base tokens are independent according to Lemma 2, two arbitrary bits in the derived token are also independent. ■

The randomness analysis of the indicators follows the same path. The simulation results provided in VIII demonstrate that the tokens and indicators have very good randomness.

G. Discussion

Memory requirement: To implement TAP, each tag needs $(u + 1)a + (v + 1)b$ bits of memory to store the keys. Our simulation results in Section VIII show that a, b, u and v can be set as small constants. Therefore, the memory requirement for the tag is small. The memory requirement at the central server is $O(n)$ for storing the key table and hash table.

Communication overhead: For each authentication, the tag only needs to transmit one a-bit token, and the reader sends an authentication request and one a reponse, both incurring $O(1)$ communication overheads.

Online computation overhead: For each authentication, the tag generates two tokens and performs one comparison to authenticate the reader. All operations performed by the tag, including bit-wise XOR, bit flip, and one-bit left circular shift, are simple and hardware efficient. The reader (or the server) needs to calculate two extra hash values: one for the token received from the tag to identify the tag, and the other for the new token to update the hash table. Both the tag and the reader have $O(1)$ computation overhead.

H. Potential Problems of TAP

TAP has three potential problems that should be addressed.

Desynchronization attack: An unauthorized reader can also initiate an authentication by sending a request. The tag will reply with its current token, and then update its keys as usual. As a result, its keys differ from what are stored by the central server. When the tag encounters a legitimate reader later, it will probably fail the authentication as its current token does not match the one stored in the central server.

Replay attack: When performing a desynchronization attack, the adversary can record the received token. Later it can retransmit the token to authenticate itself. Since the token is valid, it will pass the authentication. The above two issues also exist in the preliminary design.

Hash collision: For two tags in the system, the hash values of their current tokens may happen to be the same, called *hash collision*. In this case, their tag indexes cannot be stored in the same slot of the hash table. Otherwise, the reader cannot uniquely identify the tag through the received token. In addition, since each tag generates its tokens independently, it may happen that two tags have the same token, called *token collision*. Token collision is a special case of hash collision, and token collision must lead to hash collision. We find that hash collisions, though the probability is low, can cause problems to all anonymous RFID authentication protocols using cryptographic hashes, but the potential problems are never carefully addressed.

VI. Enhanced Dynamic Token based Authentication Protocol

To address the issues of TAP, we propose the Enhanced dynamic Token based Authentication Protocol (ETAP).

A. Resistance Against Desynchronization and Replay Attacks

Since the desynchronization attack and replay attack can be carried out simultaneously, we tack them together. Our objective is two-fold: First, the valid tag can still be successfully authenticated by a legitimate reader after some desynchronization attacks; Second, even if the adversary has captured some tokens from the valid tag, it cannot use those tokens to authenticate itself.

To make our protocol resistant against desynchronization attack, we let the central server pre-calculate an array of k tokens $[tk^1, tk^2, \ldots, tk^k]$ from the base tokens, and any token can be used to identify the tag, where k is a system parameter that can be set large or small, depending on the available memory at the server. The reader needs at least one token to identify the tag, and thus at most $k - 1$ desynchronization attacks can be tolerated[1]. After a successful mutual authentication, the reader will replenish the token array with k new tokens. Furthermore, we uses a two-step verification process to guard against replay attacks.

[1]An exponentially increasing timeout period can be enforced between unsuccessful authentications to prevent an adversary from depleting the k tokens too quickly.

Tag Index	Tag	Base Token Array	Token Array	Base Indicator Array	Indicator
1	t_1	$[bt_1^1, \ldots, bt_1^u]$	$[tk_1^1, tk_1^2, \ldots, tk_1^k]$	$[bi_1^1, \ldots, bi_1^v]$	ic_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	t_n	$[bt_n^1, \ldots, bt_n^u]$	$[tk_n^1, tk_n^2, \ldots, tk_n^k]$	$[bi_n^1, \ldots, bi_n^v]$	ic_n

TABLE III: Key table stored by the central server for ETAP.

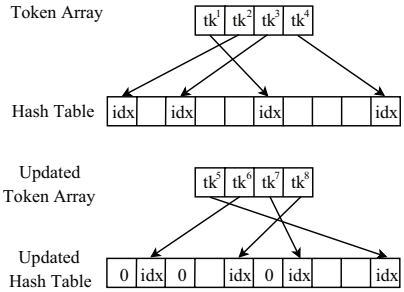

Fig. 6: Our scheme against desynchronization attack.

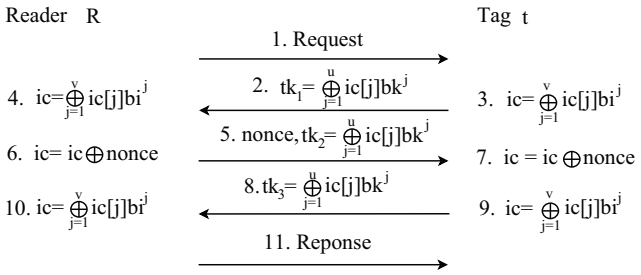

Fig. 7: Two-step verification mechanism of ETAP.

Table III shows the key table stored by the central server for implementing ETAP.

Now let us elaborate ETAP with an example given in Fig. 6. Suppose $k = 4$ and the reader pre-calculates four tokens tk^1, tk^2, tk^3 and tk^4 for tag t with tag index idx. In addition, suppose the current token stored by t is $tk = tk^2$, which means t may have been under one desynchronization attack and the adversary has captured tk^1. When the reader receives tk^2 from t, it accesses $HT[h(tk^2)]$ to fetch the tag index idx of the t, and then the token array of t from $KT[idx]$. The reader then traverses the token array until it finds tk^2. After that, the reader uses the next token in the token array, tk^3 in this example, to authenticate itself. If the received token happens to be at tail of the array, the reader needs to derive a new token for authentication. To prevent the adversary from passing authentication using tk^1, we adopt the two-step verification as illustrated in Fig. 7. In step 3, the reader includes a b-bit random nonce in its message, and challenges the tag to send another token. After the tag authenticates the reader, it updates its indicator by XORing the indicator with the received nonce (so does the reader), which contributes to randomizing the indicator as well. After that, the tag derives a new token based on the updated indicator, and sends it to the reader for the second verification. Since the adversary does not know the base tokens and the indicator, it cannot derive the correct token to pass the second verification, rendering replay attack infeasible. After the successful mutual authentication, the reader generates four new tokens to replenish the token array. In addition, the reader updates HT by setting the slots corresponding to the old tokens to 0, and setting the slots corresponding to the new tokens to idx. Note that the token replenishment is performed off line by the central server, which is therefore not a performance concern.

B. Resolving Hash Collisions

One candidate approach for reducing hash collisions is to use a very large hash table, which is however not memory efficient. We observe that two different tokens causing a hash collision under one hash function probably will not have a collision under another hash function. Therefore, using multiple hash functions provides an alternative way for resolving hash collisions. A slot in the hash table is called an *empty slot*, a *singleton slot*, and a *collision slot* respectively, if zero, one and multiple tokens are mapped to it. Suppose the size of the hash table is l, and the central server pre-computes k tokens for each tag. When a single hash function is used, the probability p_s that an arbitrary slot is a singleton slot is

$$p_s = \binom{nk}{1} \frac{1}{l} \left(1 - \frac{1}{l}\right)^{nk-1} \approx \frac{nk}{l} e^{-\frac{nk}{l}}. \quad (5)$$

It is easy to prove that $p_s \leq \frac{1}{e} \approx 0.368$ and it is maximized when $l = nk$.

We find that if we apply two independent hash functions to map tokens to slots, a slot will have a probability of up to $1 - (1 - 0.368)^2 \approx 0.601$ to be a singleton in one of the two mappings. If we apply r independent hash functions from tokens to slots, the probability that a slot will be a singleton in one of the r mappings can increase to $1 - (1 - 0.368)^r$, which quickly approaches to 1 with the increase of r. Fig. 8 presents an example showing the advantage of using multiple hash functions in reducing hash collisions. In the left plot, only one hash function is used, and there is only one singleton slot, while in the right plot, three hash functions are employed and every token is mapped to a singleton slot.

To identify which hash function maps a token of which tag to a certain singleton slot, that slot needs to store both the index of that hash function, called *hash index*, as well as the tag index. Fig. 9 shows the hash table used by ETAP. For example, a token tk_2 of t_2 is mapped by the hash function $h_3(\cdot)$ to the first slot (a singleton slot) of HT. Hence, $HT[1]$

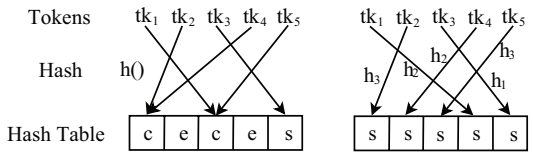

Fig. 8: An example of using multiple hash functions to reduce hash collisions, where the left plot uses one hash function, and the right plot uses three hash functions. 'e' means an empty slot, 's' means a singleton slot, and 'c' means a collision slot.

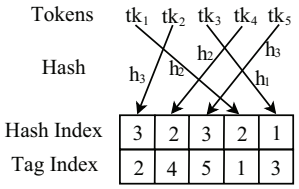

Fig. 9: A toy example of hash table used by ETAP. Each slot stores both hash index and tag index.

records the hash index 3, and the tag index 2. When the reader receives a token tk from a tag, it computes $h_i(tk)$ $(1 \leq i \leq r)$ till it finds the hash index in slot $HT[h_i(tk)]$ is equal to i, where it can obtain the correct tag index of that tag.

Our simulations results in Section VIII demonstrate that the hash collisions caused by different tokens can be resolved by using a small number of independent hash functions. However, the issue of token collisions still exists because any hash functions map the same token to the same slot. Note that if token collision happens, the reader cannot identify the tag since the token is associated with multiple tags. Therefore, such collided tokens are not useful in nature. The central server can store those tokens in a CAM (Content Addressable Memory) [30] or another hash table for quick lookup. When the reader receives a token, it first checks if it is collided one; if so, the reader needs to request another token to identify the tag. We expect that the number of collided token is small as long as the generated tokens have good randomness.

C. Discussion

Memory requirement: The memory requirement for the tag to implement ETAP is the same as TAP, i.e., $(u+1)a+(v+1)b$ bits. The memory requirement at the central server moderately increases because of the larger key table and hash table for storing multiple tokens for each tag.

Communication overhead: For each authentication, the tag only needs to transmit two a-bit tokens, and the reader needs to send an authentication request, one a-bit token, one b-bit nonce, and a response, both incurring $O(1)$ communication overheads.

Online computation overhead: For each authentication, the tag generates three tokens and performs one comparison to authenticate the reader. ETAP requires some extra computation overhead from the reader (server). First, the reader should check if a received token is a collided one, which requires $O(1)$ computation. In addition, the reader needs to calculate at most r hash values to identify the tag, and perform at most k comparisons to locate the received token in the token array. Since r and k are small constants, the online computation overhead for the reader is still $O(1)$.

Functional Block	Cost (logic gates)
2 input NAND gate	1
2 input XOR gate	2.5
2 input AND gate	2.5
FF (Flip Flop)	12
n-byte RAM	$n \times 12$

TABLE IV: Estimated costs of typical cryptographic hardware.

Hardware cost: The hardware for RFID tags to implement ETAP consists of a circular shift register, two registers for storing intermediate results, some XOR gates, and some RAM to store u base tokens, v base indicators, one token and one indicator. Table IV provides the estimated costs of typical cryptographic hardware [26], [31], which will also be used for estimating the hardware cost of ETAP. The circular shift register is a group of flip-flops connected in chain, which requires $12 \times \max(a, b)$ logic gates. Similarly, the two registers for intermediate results need $2 \times 12 \times \max(a, b)$ logic gates. In addition, it takes $2.5 \times \max(a, b)$ logic gates to implement the XOR gates. Finally, the cost of the RAM for storing the base tokens, base indicators, token and indicator is about $\frac{(u+1)a}{8} \times 12 + \frac{(v+1)b}{8} \times 12$ logic gates. Therefore, the total number of required logic gates for implementing ETAP is approximately $38.5 \times \max(a, b) + 1.5 \times (u+1)a + 1.5 \times (v+1)b$. For example, if we set $a = b = 16$, $u = 10$ and $v = 6$ (the reason for this setting will be explained shortly), ETAP only requires about 1K logic gates.

VII. Security Analysis

ETAP is designed to be resistant against desynchronization attack and replay attack. In this section, we further analyze the security of ETAP under both passive and active attacks.

Known token attack: In ETAP, the tokens are transmitted without any protection, which may lead to a potential security loophole. The adversary can capture all tokens exchanged between the reader and the tag, and use them to infer the base tokens. However, we have the following theorem:

Theorem 3. *Cracking the base tokens from the captured tokens is computationally intractable if a sufficient number of base tokens are used.*

Proof: According to (1), each captured token provides an equation of the base tokens. Since there are u base tokens, at least u independent equations are needed to obtain a solution of the base tokens. However, the adversary has no clue about the current value of the indicator, which can have very good randomness as shown in Section VIII. Therefore, the adversary has no better way than trying each possible value of the indicator by brute force. Hence, the u bits in the selector and the 2-bit update pattern give 2^{u+2} instantiations of each equation. Therefore, the adversary has

to solve $(2^{u+2})^u = 2^{u(u+2)}$ different equation sets. For each candidate solution, the adversary derives another token, and compares it with the captured one to verify if the solution is correct, which requires another 2^u trials. As a result, the total computation overhead for the adversary to crack the base tokens is $2^u \times 2^{u(u+2)} = 2^{u(u+3)}$, which is computationally intractable if u is set reasonably large, e.g., $u = 10$. ■

Anonymity: Due to the randomness of the tokens (verified in Section VIII), the adversary cannot associate any tokens with a certain tag. According to Theorem 2, the probability that the adversary can successfully guess any bit z of a tag's next token based on its previous tokens is

$$Prob(z\prime = z) \leq \frac{1}{2} + \frac{1}{ploy(s)}, \tag{6}$$

where $z\prime$ is the adversary's guess of z, and $poly(s)$ is an arbitrary polynomial with a security parameter s. Therefore, the adversary does not have a non-negligible advantage in guess z, and ETAP can preserve the anonymity of tags.

Compromising resistance: In ETAP, the keys of each tag are initialized and updated independently. Even if all tags except two are compromised by an adversary, it still cannot infer the keys of the two remaining tags or distinguish them based on their transmitted tokens. Therefore, ETAP is robust against compromising attack.

Forward secrecy: Forward secrecy requires that an adversary cannot crack the previous messages sent by a tag even if the adversary obtains the current keys of the tag. ETAP has perfect forward secrecy because in step 3 of each authentication, the tag will XOR its current indicator with a random nonce. Even if the adversary obtains all current keys of the tag, it does not know the previous values of the indicator without capturing all random nonces. Therefore, the adversary cannot perform reverse operations of the updating process to calculate the previous tokens.

VIII. Numerical Results

In this section, we first verify the effectiveness of our scheme of using multiple hash functions to resolve hash collisions. After that, we run randomness tests on the tokens generated by ETAP.

A. Effectiveness of Multi-hash Scheme

First, we determine the number l of slots needed to guarantee every token is mapped to a singleton slot when different numbers of hash functions are employed. The number of tokens is set to 100 and 1000. We vary the number r of hash functions from 1 to 10. Under each parameter setting, we repeat the simulation 500 times and obtain the average number of required slots. Results in Fig. 10 demonstrate that l is reduced dramatically with the increase of r at first, and gradually flattens out when r is sufficiently large. For example, when $r = 1$, more than 1400 slots are needed to guarantee each of the 100 tokens is mapped to singleton slot, about 14 slots per token; in contrast, when $r = 10$, the 100 tokens only require 103 slots on average, approximately 1 slot per token.

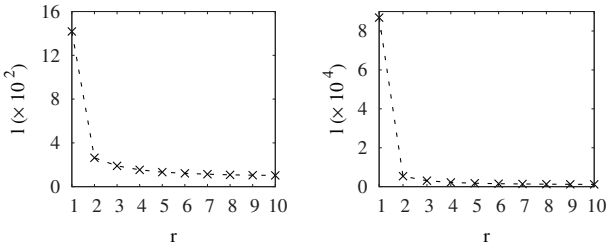

Fig. 10: Number of slots needed to guarantee every token is mapped to a singleton slot when different numbers of hash functions are used. *Left Plot*: The number of unique tokens is 100. *Right Plot*: The number of unique tokens is 1000.

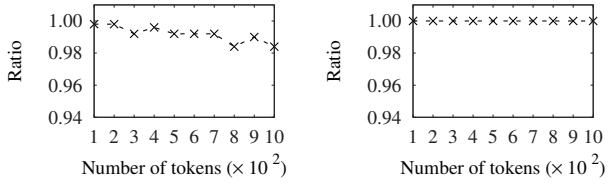

Fig. 11: Ratio of tests that have no hash collision when $r = 10$. *Left Plot*: The number of slots is 1.5× the number of tokens. *Right Plot*: The number of slots is 1.8 × the number of tokens.

Furthermore, we investigate the minimum number of slots needed to guarantee no hash collision when a fixed number hash functions are used. We fix $r = 10$, and vary the number of tokens from 100 to 1000 at steps of 100. Under each parameter setting, we run 500 tests and calculate the ratio of tests that have no hash collision. The left plot of Fig. 11 presents the results when l is set to 1.5× the number of tokens. In this case, only a few tests have hash collisions, e.g., when the number of tokens is 1000, 8 out of the 500 tests have hash collisions. When l increases to 1.8× the number of tokens, there is no hash collision any more, as shown in the right plot of Fig. 11.

B. Token-Level Randomness

The effectiveness of ETAP relies on the randomness of the tokens and indicators. An intuitive requirement of randomness is that any token (indicator) should have approximately the same probability to appear. The EPC C1G2 standard [1] specifies that for a 16-bit pseudorandom generator the probability of any 16-bit $RN16$ with value x shall be bounded by $\frac{0.8}{2^{16}} < P(RN16 = x) < \frac{1.25}{2^{16}}$. To evaluate the randomness of tokens and indicators generated by ETAP, we set $a = b =$ 16 , respectively produce $2^{16} \times 500$ tokens and indicators, and calculate the frequency of each token or indicator. Note that we can concatenate multiple tokens to form a longer one if necessary. In addition, we set $u = 10$ as suggested by Theorem 3, and vary v = 2, 4, 6 to investigate its impact on randomness. Fig. 12 presents the results, where the dotted horizontal lines represent the bounds specified by EPC C1G2. We can see that the indicators have better randomness with the increase of v,

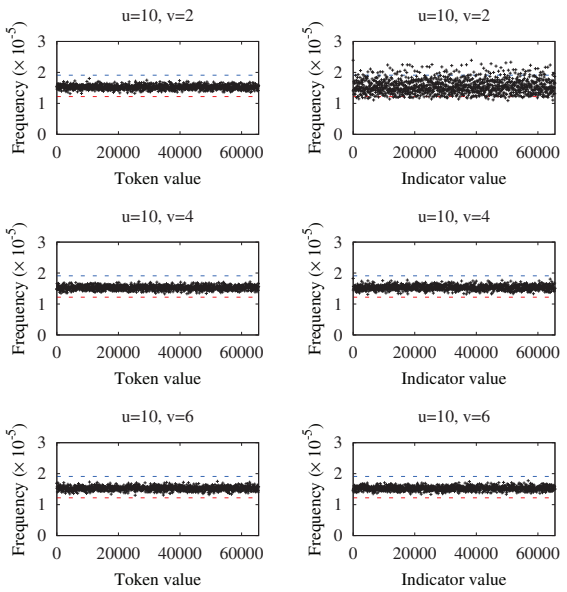

Fig. 12: Frequency tests for tokens and indicators generated by ETAP, where $a = b = 16$. Each point represents a token/indicator and its frequency, and the two dotted horizontal lines represent the required bounds.

while the randomness of tokens is not sensitive to the value of v since u is already set sufficiently large. In addition, when $u = 10$ and $v = 4$, requiring only 256-bit tag memory, both the tokens and indicators meet the randomness requirement.

C. Bit-Level Randomness

The National Institute of Standards and Technology (NIST) provides a statistical suite for randomness test [32], including monobit frequency test, block frequency test, cumulative sums test, runs test, test for the longest run of ones in a block, binary matrix rank test, etc. Due to space limitation, we cannot explain each test here, and interested readers can refer to [32] for detail information. Given a sequence of n_s bits, it is accepted as random only if the observed p-value is no less than a pre-specified *level of significance* α based on the null hypothesis H_0.

We use two metrics to evaluate the test results: (1) The proportion of sequences that pass the tests. The acceptable range is $\hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m_s}}$ [32], where $\hat{p} = 1 - \alpha$ and m_s is the sample size; (2) The uniformity of the observed p-values. Let X be a random variable with probability density function $f_X(x)$, and $Y \in [0, 1]$ be the p-value of X. Since the cumulative distribution function $F(X)$ of X is monotonically increasing, we have

$$\begin{aligned} P(Y \le y) &= P(\int_X^{\infty} f_X(x)dx \le y) = P(1 - F(X) \le y) \\ &= 1 - P(X \le F^{-1}(1-y)) = 1 - (1-y) = y. \end{aligned} \tag{7}$$

[2]Matrix rank test requires that the bit sequence consists of at least 38912 bits. Hence, no test is performed when $n_s < 38912$, which is marked as N.A..

Hence, $Y \sim U(0, 1)$. We divide $(0, 1)$ into ten equal-length subintervals, and denote the numbers of p-values in each subinterval as F_1, F_2, ..., F_{10}, respectively. We have

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - \frac{m_s}{10})^2}{\frac{9m_s}{100}} \sim \chi^2(9). \tag{8}$$

The proof is given in the Appendix. Therefore, we can employ χ^2 test. If the observed statistic of χ^2 is $\chi^2(obs)$, the p-value is

$$\begin{aligned} P(\chi^2 \ge \chi^2_{obs}) &= \frac{\int_{\chi^2(obs)}^{\infty} e^{-x/2} x^{9/2-1} dx}{\Gamma(9/2)2^{9/2}} \\ &= \frac{\int_{\chi^2(obs)/2}^{\infty} e^{-x} x^{9/2-1} dx}{\Gamma(9/2)} \\ &= igamc(\frac{9}{2}, \frac{\chi^2(obs)}{2}), \end{aligned}$$

where $igamc\,(c, z) = 1 - \frac{\int_{-\infty}^{z} e^{-x} x^{c-1} dx}{\Gamma(c)}$. The uniformity is acceptable if $igamc(\frac{9}{2}, \frac{\chi^2(obs)}{2}) \ge 0.0001$ [32].

We set $a = b = 16$, $u = 10$ and $v = 4$, and convert the tokens generated by ETAP to a bit sequence. We vary n_s from 1000, 5000, 10000 to 50000. The NIST suggests that $\alpha \ge 0.001$, so we set $\alpha = 0.01$. In addition, we set $m_s = 500$, in the same order of magnitude as α^{-1}. The block size M should be selected such that $M \ge 20$, $M > 0.01n_s$ and $N_B < 100$, where N_B is the number of blocks. We set $M = 0.02n_s$, so $N_B = \frac{n_s}{M} = 50$.

The test results are shown in Table V. We can see that the bit sequence generated by ETAP can pass the randomness tests under all parameter settings, which again verifies that our protocol can generate tokens with good randomness.

IX. Conclusion

In this paper, we propose a lightweight anonymous authentication protocol for RFID systems. To meet the hardware constraint of low-cost tags, we abandon hardware-intensive cryptographic hashes and follow the asymmetry design principle. Our protocol ETAP uses a novel technique to generate random tokens on demand for anonymous authentication. The randomness analysis and tests demonstrate that ETAP can produce tokens with very good randomness. Moreover, ETAP reduces the communication overhead and online computation overhead to $O(1)$ per authentication for both the tags and the readers, which compares favorably with the prior art.

X. Acknowledgments

This work was supported in part by the National Science Foundation under grant NeTS-1409797, and a grant from Florida Cybersecurity Center.

References

[1] *EPC Radio-Frequency Identity Protocols Class-1 Gen-2 UHF RFID Protocol for Communications at 860MHz-960MHz, EPCglobal*, available at http://www.gs1.org/sites/default/files/docs/epc/uhfc1g2_1_2_0-standard-20080511.pdf, Version 1.2.0.

Test \ Sequence Length	1000		5000		10000		50000	
	Proportion	p-value	Proportion	p-value	Proportion	p-value	Proportion	p-value
Monobit Frequency	0.9920	0.002927	0.9880	0.861264	0.9920	0.719747	0.9900	0.957612
Block Frequency	0.9960	0.037076	0.9880	0.538182	0.9880	0.162606	0.9940	0.055361
Cumulative Sum(Mode 0)	0.9920	0.119508	0.9860	0.123755	0.9880	0.632955	0.9880	0.986227
Cumulative Sum(Mode 1)	0.9920	0.798139	0.9920	0.823725	0.9900	0.877083	0.9900	0.081510
Runs	0.9940	0.286836	0.9820	0.482707	0.9880	0.146982	0.9860	0.068571
Longest Run	0.9960	0.583145	0.9880	0.554420	0.9860	0.851383	0.9900	0.889118
Matrix Rank[2]	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	0.9880	0.004697

TABLE V: The sample size m_s is 500, and the acceptable confidence interval of the success proportion is $[0.97665, 1]$.

[2] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk, "Efficient and Reliable Low-power Backscatter Networks," *Proc. of ACM SIGCOMM*, 2012.

[3] M. Trotter, C. Valenta, G. Koo, B. Marshall, and G. Durgin, "Multi-Antenna Techniques for Enabling Passive RFID Tags and Sensors at Microwave Frequencies," *Proc. of IEEE RFID*, 2012.

[4] J. Liu, B. Xiao, K. Bu, and L.Chen, "Efficient Distributed Query Processing in Large RFID-enabled Supply Chains," *Proc. of IEEE INFOCOM*, 2014.

[5] K. Bu, B. Xiao, Q. Xiao, and S. Chen, "Efficient Pinpointing of Misplaced Tags in Large RFID Systems," *Proc. of IEEE SECON*, pp. 287–295, 2011.

[6] H. Yue, C. Zhang, M. Pan, Y. Fang, and S. Chen, "A Time-efficient Information Collection Protocol for Large-scale RFID Systems," *Proc. of IEEE INFOCOM*, pp. 2158–2166, 2012.

[7] Q. Xiao, M. Chen, S. Chen, and Y. Zhou, "Temporally or Spatially Dispersed Joint RFID Estimation Using Snapshots of Variable Lengths," *Proc. of ACM Mobihoc*, 2015.

[8] W. Luo, Y. Qiao, and S. Chen, "An Efficient Protocol for RFID Multigroup Threshold-based Classification," *Proc. of IEEE INFOCOM*, April 2013.

[9] M. Chen, W. Luo, Z. Mo, S. Chen, and Y. Fang, "An Efficient Tag Search Protocol in Large-Scale RFID Systems," *Proc. of IEEE INFOCOM*, 2013.

[10] *RFID Report*, http://www.idtechex.com/research/reports/rfid-forecasts-players-and-opportunities-2014-2024-000368.asp, 2013.

[11] *Sunpass*, available at http://en.wikipedia.org/wiki/SunPass.

[12] *SINIAV*, available at http://roadpricing.blogspot.com/2012/08/brazil-to-have-compulsory-toll-tags-by.html.

[13] A. Bogdanov, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, and Y. Seurin, "Hash Functions and RFID Tags: Mind the Gap," *Proc. of CHES*, pp. 283 – 299, 2008.

[14] M. Chen and S. Chen, "An Efficient Anonymous Authentication Protocol for RFID Systems Using Dynamic Tokens," *Proc. of IEEE ICDCS*, 2015.

[15] S. Weis, S. Sarma, R. Rivest, and D. Engels, "Security and Privacy Aspects of Low-cost Radio Frequency Identification Systems," *Lecture notes in Computer Science*, 2004.

[16] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Efficient Hash-chain based RFID Privacy Protection Scheme," *ICUCU, Workshop Privacy*, 2004.

[17] G. Avoine and P. Oechslin, "A Scalable and Provably Secure Hash-based RFID Protocol," *IEEE PerCom Workshops*, pp. 110 – 114, March 2005.

[18] G. Avoine, E. Dysli, and P. Oechslin, "Reducing Time Complexity in RFID Systems," *Selected Areas in Cryptography*, pp. 291–306, 2006.

[19] G. Tsudik, "Ya-trap: Yet Another Rrivial RFID Authentication Protocol," *Proc. of IEEE PerCom*, 2006.

[20] A. Juels and S. A. Weis, "Defining Strong Privacy for RFID," *IEEE PerCom Workshops*, pp. 342 – 347, March 2007.

[21] L. Lu, Y. Liu, and X. Li, "Refresh: Weak Privacy Model for RFID Systems," *Proc. of IEEE INFOCOM*, 2010.

[22] T. Dimitriou, "A Secure and Efficient RFID Protocol that could make Big Brother (partially) Obsolete," *Proc. of IEEE PERCOM*, 2006.

[23] L. Lu, J. Han, L. Hu, Y. Liu, and L. Ni, "Dynamic Key-Updating: Privacy-Preserving Authentication for RFID Systems," *Proc. of IEEE PERCOM*, 2007.

[24] L. Lu, J. Han, R. Xiao, and Y. Liu, "ACTION: Breaking the Privacy Barrier for RFID Systems," *Proc. of IEEE INFOCOM*, 2009.

[25] T. Li, W. Luo, Z. Mo, and S. Chen, "Privacy-preserving RFID Authentication based on Cryptographical Encoding," *Proc. of IEEE INFOCOM*, 2012.

[26] D. C. Ranasinghe and P. H. Cole, *Networked RFID Systems and Lightweight Cryptography, Chapter 8 An Evaluation Framework.* Springer Berlin Heidelberg, November 2008.

[27] *High Memory On Metal UHF RFID Tags*, available at http://www.oatsystems.com/OAT_Xerafy_RFID_Aerospace_2013/media/High-Memory-Tag-Guide.pdf.

[28] S. Pais and J. Symonds, "Data Storage on a RFID Tag for a Distributed System," *International Journal of UbiComp*, vol. 2, no. 2, April 2011.

[29] *Singular Value Decomposition*, available at http://en.wikipedia.org/wiki/Singular_value_decomposition.

[30] L. Chisvin and R. J. Duckworth, "Content-addressable and Associative Memory: Alternatives to the Ubiquitous RAM," *IEEE Computer*, vol. 22, pp. 51–64, July 1989.

[31] M. Chen, S. Chen, and Q. Xiao, "Pandaka: A Lightweight Cipher for RFID Systems," *Proc. of IEEE INFOCOM*, April-May 2014.

[32] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, and L. E. B. III, "A statistical test suite for random and pseudorandom number generators for cryptographic applications ," *National Institute of Standards and Technology*, 2010.

APPENDIX
PROOF OF (8)

Consider the value of F_i, where $1 \leq i \leq 10$. Let Z_{ij} be the event that the $p-$value of the jth ($1 \leq j \leq m_s$) sequence, denoted by Y_j, belongs to the ith subinterval $[\frac{i-1}{10}, \frac{i}{10})$. In addition, let $1_{Z_{ij}}$ be the corresponding indicator random variable, namely,

$$1_{Z_{ij}} = \begin{cases} 1, & \text{if } Y_j \in [\frac{i-1}{10}, \frac{i}{10}), \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, we have $F_i = \sum_{j=1}^{m_s} 1_{Z_{ij}}$. Since $Y_j \sim U(0,1)$, we have $E(1_{Z_{ij}}) = \frac{1}{10}$ and $Var(1_{Z_{ij}}) = \frac{1}{10} \times (1 - \frac{1}{10}) = \frac{9}{100}$. Hence, $E(F_i) = \frac{m_s}{10}$, and $Var(F_i) = \frac{9m_s}{100}$. Based on the Central Limit Theorem (CLT), $\frac{F_i}{m_s}$ converges to $Norm(\frac{1}{10}, \frac{9}{100m_s})$ asymptotically. Therefore, $(\frac{\frac{F_i}{m_s} - \frac{1}{10}}{\frac{3}{10\sqrt{m_s}}})^2 \sim \chi^2(1)$, and $\chi^2 = \sum_{i=1}^{10} (\frac{\frac{F_i}{m_s} - \frac{1}{10}}{\frac{3}{10\sqrt{m_s}}})^2 = \sum_{i=1}^{10} \frac{(F_i - \frac{m_s}{10})^2}{\frac{9m_s}{100}} \sim \chi^2(9)$.