TRANSACTIONS ON EMERGING TELECOMMUNICATIONS TECHNOLOGIES
Trans. Emerging Tel. Tech. (2013)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/ett.2624

RESEARCH ARTICLE

A constant-time identifying large-scale RFID tags using lines on a plane

Jue-Sam Chou*

Department of Information Management, Nanhua University, Chiayi 622, Taiwan

ABSTRACT

In this paper, we propose a new approach to identify a tag of a radio frequency identification system in constant time while keeping untraceability to the tag. Our scheme does not use any cryptographic primitives. Instead, we use a line in a plane to represent a tag. The points on the line, which are infinite and different each other, can be used as tag identification. We also explore the scalability of the proposed scheme. The result of experiments showed that a tag of the radio frequency identification system over 1 000 000 tags, embedded 3 K memory, can store 559 dynamic identity proofs. Copyright © 2013 John Wiley & Sons, Ltd.

KEY WORDS
radio frequency identification RFID; security and privacy; untraceability; identification protocol; location privacy; forward privacy; scalability

***Correspondence**
Jue-Sam Chou, Department of Information Management, Nanhua University, Chiayi 622, Taiwan.
E-mail: jschou@mail.nhu.edu.tw

Received 10 September 2012; Revised 15 November 2012; Accepted 18 January 2013

1. INTRODUCTION

The radio frequency identification (RFID) technique allows identifying hundreds of objects one time via a contactless manner. It therefore becomes an important role in many applications such as automobile immobilisation, real-time location systems, baggage handling, animal tracing, and item-level tagging in fashion apparels. However, the RFID technique brings not only new opportunities but also new challenges. In particular, secure and private RFID tag identification protocol is a demanding task because the resource of RFID tags is extremely limited. There are three roles in a typical RFID system: *tags* that are embedded in objects to be identified, *readers* that emit radio signals to interrogate tags and a server that maintains all tags' information, identifies tags and provides services. On the basis of the resource limitation and the operations supported on tags, RFID protocol can be divided into four classes: (i) *full-fledged*, supporting encryption or public key algorithms [1–6], (ii) *simple*, supporting pseudo random number generator and hash functions [7–18], (iii) *lightweight*, supporting pseudo random number generator and checksum functions [19], and (iv) *ultralightweight*, supporting only bitwise operations [20–22]. In this paper, our approach makes tags compute nothing but only fetching memory, and the resource cost for such tags is estimated as the cost in simple class. In addition, we examine three features of the RFID identification protocols in this class.

(1) *Low-cost limitation*: A passive RFID tag is not powered and accommodates only a few hundreds to thousands gates. Traditional cryptographic primitives are thus hardly applied on such cheap tags [7, 8, 23].
(2) *Location privacy concern*: As tags are usually embedded in objects carried by people everywhere, the user *location privacy* is an essential requirement [7, 8, 24–26]. A common countermeasure is a tag answers a server with a dynamic identity (DID). The server then solves the DID and extract the real tag identity. Meanwhile, a third party cannot link the DID to any particular tag and thus cannot locate the user who carries the tag. Researchers also refer this property as *untraceability* or *unlinkability*.
(3) *Scalability (constant-time identification)*: Because many RFID applications require deploying the tags in large scale, the scalability is also an important feature. If a server takes linear time to identify a tag, then the identification time for the server will

increase as the number of tags increase. This will limit the scalability of an RFID system. Conversely, if a server takes constant time to identify a tag, then the number of tags will not be limited. Therefore, RFID scalability can be realised as constant-time tag identification.

1.1. Related works

In recent decades, many secure and private RFID Authentication Protocol are proposed. Weies *et al.* [7] proposed a hash-based method to reserve user location privacy. Their scheme uses a random number r and tag's secret key k to make a DID, that is, $h(k, r)$, where $h(.)$ is a one-way hash function. However, the server has to linearly search its database (DB) to compare whether each $h(k_i, r)$ is equal to the received $h(k, r)$. This limits the scalability. Recent work [9] is a similar approach and suffers the same problem.

Ohkubo, Suzuki and Kinoshita [8] proposed another hashed-based method to further assure *forward secrecy*: even if tag's secrecy is exposed, the past transactions that the tag was involved cannot be linked to the tag. In their scheme, a tag and server share secrecy s_0 when initialized. After deployment, on receiving the ith query, the tag responds with $h(s_i)$, where $s_i = g(s_{i-1})$ and $g(.)$ is another one-way hash function. On seeing $h(s_i)$, the server reads its DB record by record. Suppose it is reading the jth record and s_j^* is the corresponding seed, the server will compare whether the received $h(s_i)$ is equal to $h\left(g^1\left(s_j^*\right)\right), h\left(g^2\left(s_j^*\right)\right), \ldots,$ or $h\left(g^m\left(s_j^*\right)\right)$, where $g^m(.)$ indicates performing function $g(.)$ m times. Consequently, the server has to take O(mN) time for each tag identification, where N refers to the number of tags.

Studies [10, 11] arrange tags in a tree structure on secret keys they possess to reduce the identification complexity to O(log$N$). However, the scheme [10] will be broken because compromising 20 tags in a system of $N = 2^{20}$ tags reveals the identities of other tags. The improved scheme [11] requires updating overhead in O(log$N$) to remedy the problem in [10]. Another major weakness is that the tree-based approach leads high communication overhead between tag and reader. Some researchers believe these drawbacks overweigh the reduction in identification complexity [16].

RFID Authentication Protocol series [12–15] use monotonically increasing time as the randomness of a hash-based DID. In an identification process for a group of tags, a server first computes each expected DID by using a new challenge time and then integrates tags using the challenge. On receiving an expected DID, the server can directly address the corresponding tag. This design assures untraceability and resists replay attack. It can achieve O(1) time to identify a tag for best case but still O(N) times for worst cases. In the worst case, a tag, suffered malicious queries before, will not answer the expected response, and the server will therefore launch brute searching for the tag identification.

Alomair *et al.* [16, 17] use two layers pointers to save pre-built DIDs—$h(\Psi_i, c)$s, where Ψ_i is a pseudonym and c is the counter ranging form zero to C—to allow malicious queries to a specific tag at least C times. The size of the first-layer table is estimated as O(NC), and each record point t o a second-layer table. Each second-layer table that points to a tag information is expected to contain only one record. As a result, it assures constant-time identification when server seeing $h(\Psi_i, c)$. After recognising Ψ_i, the server assigns an unoccupied pseudonym Ψ_k to the tag, where $h(\Psi_k, 0)$, $h(\Psi_k, 1), \ldots,$ and $h(\Psi_k, C)$ in the second-layer tables point to an empty tag record, position p. Finally, the server updates its tables by moving the tag information record to the new position p and emptying the original tag record, say position p', as null. By this way, it assures O(1) update.

Ryu and Takagi [27] store one-time values $\Delta = \{\alpha_1, \ldots, \alpha_m\}$ as DIDs on a tag where $\alpha_i = E_{pk}(TagID||r)$, $E_{pk}(.)$ is a public key encryption and the corresponding private key only known to the server. For each interrogation, the tag responds with a fresh one-time pad, and a reader is allowed to write new one-time values in the tag after successfully mutual authentication. As the server can obtain tag identity by decrypting a received α_i, Ryu and Takagi achieve constant-time identification and thus assure scalability. However, for about 60-bit security level, the size of α_i is about 512 bits in Rivest, Shamir and Adleman encryption or 400 bits in ElGamal elliptic-curve encryption (ECC). Then, suppose a tag with 3 K bytes memory. It can store only 48 RSA-based or 61 ECC-based ciphertexts. This implies the tolerance for malicious query to a tag is only 48 or 61 times.

Studies [1–3] use ECC-based approach, whereas studies [4, 5] use Rabin's cryptosystem, which are public-key based approaches. Public key cryptosystems easily obtain constant-time tag identification, but they pay hardware cost at tag side. The most efficient ECC component costs 12.5 K gates [28], whereas 512-bit Rabin's encryption costs about 17 K gates [29] or 10 K gates [30]. These obviously greatly exceed the cost of hash-like component Advanced Encryption Standard, about 3.4 K gates [31].

In this paper, we propose a new approach that is extremely low cost on tags while assuring constant-time tag identification and keeping tag untraceability. Our scheme uses the points of a line on a plan as one-time pseudonyms for a tag whose identity is the slope of the line. In an initialisation phase, a server transforms m points of a line into a smaller space and then assigns the transformed data (playing a role like DID) to a tag. In an identification phase, a tag answers a fresh DID; the server then reverses the DID into a point to compute the original line and therefore identifies the tag in constant time. Table I lists resource usage of the proposed scheme and previous related works.

1.2. Organisation

The remaining of this paper is organised as follows. Section 2 describes the basic idea of the proposed scheme,

Table I. Resource usage of the proposed scheme and previous related works.

Schemes	DB size	Identification	DB update	Tag space	Communication	Note
Weis [7, 9]	O(N)	O(N)	None	O(1)	O(1)	
OSK [8]	O(N)	O(mN)	O(1)	O(1)	O(1)	
Tree-based [10, 11]	O(N)	O(lg$N$)	O(lg$N$)	O(1)	O(lg$N$)	
RAP series [12–15]	O(N)	O(1) or O(N)	O(1)	O(1)	O(1)	
Alomair [16, 17]	O(NC)	O(1)	O(1)	O(1)	O(1)	
RT [27]	O(N)	O(1)	None	O(m)	O(1)	m is limited about 40–60
PKC-based [1–5]	O(N)	O(1)	None	O(1)	O(1)	Tag costs more gates
Ours	O(N)	O(1)	None	O(m)	O(1)	m is limited about 400–500

OSK, Ohkubo, Suzuki and Kinoshita; DB, database; RT, Ryu and Takagi; RAP, RFID Authentication Protocol; PKC, Public Key Cryptosystem.

whereas Section 3 presents the detailed implementation. Performance evaluation and security analyses of the proposed scheme are shown in Sections 4 and 5, respectively. Finally, Section 6 gives conclusion and future work.

2. BASIC IDEA

In this work, we would like to use the slope of a line on a plane to represent a tag. The tag then can be identified if it can provide correct information regarding the line. Figure 1 shows the basic idea of our scheme. In the figure, a randomly chosen point (a, b) is the secrecy of the server. For simplification and without loss of generality, we let $a = b = 0$. A line L_i, having slope s_i and passing through (a, b), represents tag T_i. Thus, any point (x, y) on the line L_i can be a proof of T_i. Moreover, we assume that the length of a, b and x each is k-bit long, denoted as $|a| = |b| = |x| = k$. In addition, we define the set $S = \{s \mid s \text{ is an integer}, -(2^{k-1} - 1) \leqslant s \leqslant +(2^{k-1} - 1), \text{ and } s \neq 0\}$ to be slopes of lines representing all tags of an RFID system. Hence, the number of tags in our system is $N = 2^k - 2$. The following algorithm demonstrates the tag initialization of this basic scheme.

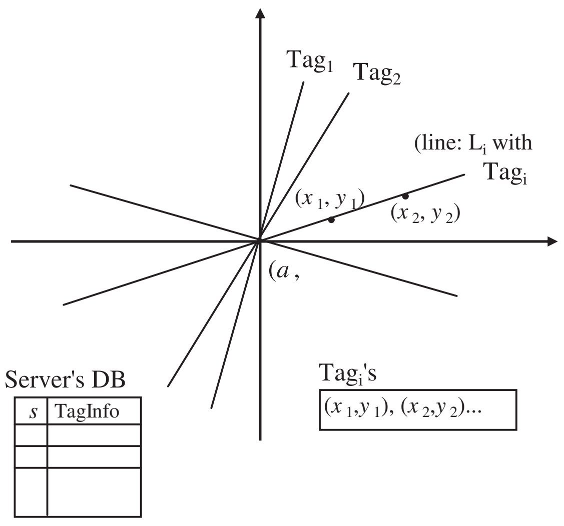


Figure 1. Basic model of the study. DB, database.

```
Basic Tag Initialization {
    $S = [-(2^{k-1}-1), +(2^{k-1}-1)] - \{0\}$;
    $N = |S|$; $i = 1$;                          //No. of tags
    For $s = -2^{k-1}+1$ to $2^{k-1}-1$   {     //s represents the slope of a line
        setup tag $T_i$;
        for $j = 1$ to $m$ {                    // generate m pairs of (x, y)
            $x_j \leftarrow_R [-(2^{k-1}-1), +(2^{k-1}-1)] - \{a\}$;
            $y_j = s * x_j + (b - a * s)$;
        }
        Write m tuples, $(x_1, y_1), \ldots, (x_m, y_m)$, into $T_i$'s storage;
        Insert $\{s, \text{TagInfo}, x_1, x_2, \ldots, x_m\}$ into server's database;
        Next i;
    }
}
```

When the tag initialization completed, each tag storage has m tuples of proofs, (x_1, y_1), $(x_2, y_2), \ldots,$ and (x_m, y_m). In the identification phase, on receiving an interrogation from a server, a tag will answer an unused (x_j, y_j) pair. On receiving the pair, the server will compute $s = (x_j - a)/(y_j - b)$ and then use s to look up the corresponding tag in its DB.

We take $k = 4$ as an example. The set S is all integers in $[-7, 7] - \{0\}$, $|S| = N = 14$, and each element in S represents an identity of a tag. In addition, let a random pair $(a, b) = (-5, 7)$ be the secret point of the server. Figure 2 illustrates 14 lines of the tags. For example, $T_{11} : y = 4x + 27$, having slop four and passing $(-5, 7)$, represents a tag with identity four. The points (5, 47), (1, 31) and (3, 39) on line T_{11} are the proofs of the tag. In an identification process, the tag can provide any one of these proofs, say (5, 47). The server will be able to compute the slop, $s = (47-7)/(5-(-5)) = 4$, to obtain the identity of the tag.

Our approach has several merits: First, it assures constant-time tag lookup at server side because a server can compute the slope of a tag and then directly fetch the tag record from server's DB. Furthermore, the constant-time tag identification implies the *scalability* of an RFID system when compared with O(N)-time tag identification of many hash-based schemes [3, 4, 8, 11–14] (also see Table I). The second merit of our approach is that it

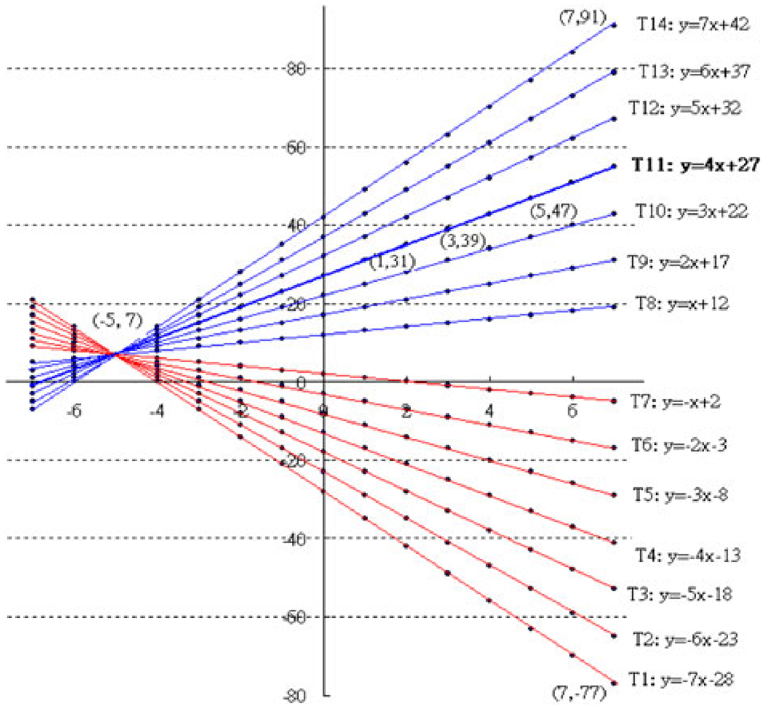


Figure 2. Example of 4-bit radio frequency identification systems in basic model.

requires only O(N) space to store tag information at server side. Thirdly, the tag need not embed any cryptographic component such as Pseudo Random Number Generator (PRNG), Hash or AES. When being identified, the tag just selects an unused proof as its answer. This takes extremely low computational overhead.

However, some details should be further considered. *First*, the plaintext of point (x, y) is not suitable for transmission over an open network because any two eavesdropped pairs (x_1, y_1) and (x_2, y_2) from a tag can deduce the slope of the tag. Therefore, (x, y) pair should be transmitted in an encrypted form. Our countermeasure of this paper is that the server transforms a (x, y) into a random-looking string in the initialization phase. Thus, the random-looking result will play as a one-time pad and hence achieve *unconditional security*. In other words, even an adversary intends to exhaustively guess the server's secret point (a, b), he or she does not has any clue—the plain (x, y)s or slopes—to examine his or her guess. *Second*, the tag's memory size will be a limitation. As x and s each is a k-bit integer, the value of the y will be about $2k$-bit long. (This can be easily observed. For example, in the case of Figure 2, the value y ranges from -77 to 91 when $(a, b) =$ (-5, 7) and x ranges from -7 to 7. Directly storing such y value needs 8 bits, whereas storing x value only takes 4 bits.) If a tag stores such an (x, y) pair in a plain manner, then it would take $3k$ bits. In other words, supposing k is 20 bits, the tag would take 60 bits of storage to store an (x, y) pair. Thus, a tag having 2 K-byte storage can store about 273 pairs of (x, y). We consider that if the storage size of the y can be reduced to 20 bits as same as the storage size of x, the tag then can store over 400 pairs of (x, y). Therefore, the problem of this study can be further formulated as how to downsize the storage of the y values.

3. THE PROPOSED SCHEME

In this section, we present a practical implementation on the basis of our basic model. In the implementation, we first consider how to store the y values in tags; that is, how to encode the y value to attain better space efficiency.

We start the discussion from the range of y values. The maximum of all possible y for any (a, b), denoted as YMAX, is

$$\text{YMAX} = 2(2^{k-1} - 1)^2 - 3{*}2^{k-1} + 1 \approx 2^{2k-1}$$

when $(s, a, b, x) = (2^{k-1} - 1, -(2^{k-1} - 1), 2^{k-1} - 1, 2^{k-1} - 1)$ or $(-(2^{k-1} - 1), 2^{k-1} - 1, 2^{k-1} - 1, -(2^{k-1} - 1))$. And y will reach the minimum, denoted as YMIN,

$$\text{YMIN} = -2(2^{k-1} - 1)^2 + 3{*}2^{k-1} - 1 \approx -2^{2k-1}$$

when $(s, a, b, x) = (-(2^{k-1} - 1), -(2^{k-1} - 1), -(2^{k-1} - 1), 2^{k-1} - 1)$ or $(2^{k-1} - 1, 2^{k-1} - 1, -(2^{k-1} - 1), -(2^{k-1} - 1))$.

Take k as 4 bits for example. The range of x will fall in $[-7, 7]$, whereas the range of y will fall in $[-105, 105]$. YMAX is 105 when $(s, a, b, x) = (7, -7, 7, 7)$ or $(-7, 7, 7, -7)$, whereas YMIN is -105 when $(s, a, b, x) = (-7, -7, -7, 7)$ or $(7, 7, -7, -7)$. Thus, storing y value requires about 8 bits, whereas storing x value requires only 4 bits.

However, according to our observation, the distribution of y is not uniform; it is sparse in the space [YMAX, YMIN]; especially, when y is toward extreme values, it is sparser. The example ($k = 4, (a, b) = (-5, 7)$) shown in Figure 2 also supports this observation. In the case, the y ranges from $[-77, 91]$; the space size of $[-77, 91]$ is 169. However, the number of occurrence of the different y values is only 90. Among them, the number of occurrence of the different y values that fall in $[-34, 49]$, the middle range of $[-77, 91]$, is 59, occupied $59/90 = 66\%$. Consequently, we believe that y value can be compressed through some transformation algorithms and mapped to a smaller space. Our intuitive solution has two phases.

First, we define set Y as all y values yielded by the basic tag initialization algorithm, where y should range from -2^{2k-1} to 2^{2k-1} if $|x| = k$ bits. In this phase, we want to fuzzifies y values in the Y to reduce the order (cardinality) of Y. A value $y^{(1)}$ in Y can be fuzzified as $y^{(2)}$ if

(1) $y^{(2)}$ is an element of Y,
(2) $y^{(2)}$ is proximate to $y^{(1)}$, and
(3) the slope of the line passing through the fuzzified point (whose y-axis is $y^{(2)}$) should fall between $s + 0.5$ and $s - 0.5$, where s is the slope of the line passing the original point (whose y-axis is $y^{(1)}$).

After fuzzification, the set Y is reduced to the set Y', where $|Y'|$ should be smaller than $|Y|$.

Second, we use a table to map each y' in Y' to a distinct random number y'', that is, $MAP : \{0, 1\}^{2k} \rightarrow \{0, 1\}^{k''}$, yielding Y'', where k'' is a properly selected value.

The following algorithm presents the aforementioned two-phase transformation, and Figure 3 shows its corresponding flow charts.

Tag Initialization Algorithm (a, b)

```
{
  $S = [-(2^{k-1}-1), (2^{k-1}-1)] - \{0\}$;
  $N = |S|$;                        //No. of tags
  $s = -2^{k-1}+1$;                 //$s$ represents the slope of a line
  for $i = 1$ to $N$
      // Setup tag $T_i$;
      for $j = 1$ to $m$ {          // produce $m$ pairs of $(x, y)$
          $x_j \leftarrow_R S \cup \{0\} - \{a\}$;
          $y_j = s * x_j + (b - a * s)$;
          $y''_j$ = Transformation($y_j$);
      }
      Write $m$ tuples $(x_1, y''_1), \ldots, (x_m, y''_m)$ into $T_i$'s storage;
      Insert $\{s$, TagInfo, $x_1, x_2, \ldots, x_m\}$ into database;
      Next $s$ in $S$;
}
Transformation($y$)   {
    List $ylist$;   //each element $y^*$ in $ylist$ has three fields: $yVal$, $count$, $randomNo$;
                    //initialized with empty;
    Search $y$'s proximate value $y^*$ in the $ylist$;
    If found   {
        $slope^* = (y^* - b)/(x - a)$;
        If $|slope^* - slope| < 0.5$   {
            $y^*.count{+}{+}$;
            Return $y^*.randomNo$;
        }
    }
    New $y^*$;
    $y^*.yVal = y$;
    $y^*.count = 1$;
    $y^*.randomNo \leftarrow \{0, 1\}^{k''}$;
    Insert $y^*$ into the $ylist$;
    Return $y^*.randomNo$;
}
```

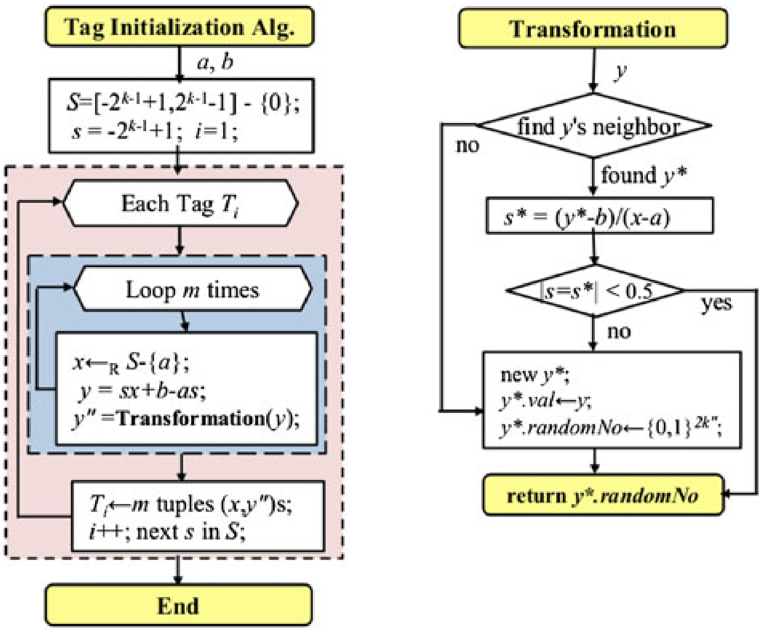


Figure 3. Flow charts of tag initialization algorithm.

In the identification phase, the server/reader interrogates tags with message 'hello'. On receiving a valid interrogation message, a tag answers a fresh (x, y'') pair and sets the memory position of the pair by null value. The server then performs *tag identification algorithm* to identify the tag as follows.

```
Tag Identification Algorithm (x, y")
{
    y* ← Search ylist using y".
    slope* = (y* - b)/ (x- a);
    slope = FindProximateInteger(slope*);
    Read tag record using slope;
    If not found return "Invalid Data";
    If x is not fresh return "Replay";
    Else return "Accept";
}
FindProximateInteger(s)
{
    If (s–Floor(s) < 0.5)
        Return Floor(s);
    Else
        Return Ceiling(s);
}
```

Example. *Take $k = 8$ as an example and suppose $(a, b) = (-23, -94)$. The tag initialization algorithm starts from $s = -127$ to $s = -127$. When $s = -127$, it sets up tag T_1 and assigns its identity as -127, that is, $T_1 : y = -127x - 3015$. The algorithm continues generating the first x value randomly, say 49, and computes the corresponding $y = -9238$. Then, it transforms the value y. As the ylist is empty now, the algorithm inserts $y = -9238$ into the ylist and returns a random string from $[0, 2^{11}]$, say 1102. Thus, the first proof (49, 1102) for T_1 is generated. The algorithm continues generating the second random x, say 36, and computes corresponding $y = 1557$. Similarly, to transform the value y, the algorithm finds the adjacent integer in the ylist, say -9238, and checks whether the slope $s*$ of a line, passing through (a, b) and $(36, -9238)$, falls between $(-127 - 0.5, -127 + 0.5)$. Because it does not, the algorithm inserts the new $y = 1157$ into the ylist and returns a random string v, say 25. Again, the second proof (36, 25) for T_1 is generated. The algorithm goes on such steps m times and finally produces m proofs for tag T_1. Now, suppose that the algorithm comes to $s = -35$ and it sets up $T_{93} : y = -35x - 899$. The first random $x = -70$ and corresponding $y = 1551$. In transformation, the algorithm finds an adjacent 1557 in the ylist and computes the slope s^* of a line passing through (a, b) and $(-70, 1557)$ is equal to $(1557-(-94))/(-70-(-23)) = -35.13$, falling in $(-35 - 0.5, -35 + 0.5)$. It then uses 1557 instead of 1551. So, the proof of the line $(s = -35)$ becomes $(-70, 25)$, where the value 25 is the corresponding random number of the value 1577 in the ylist.*

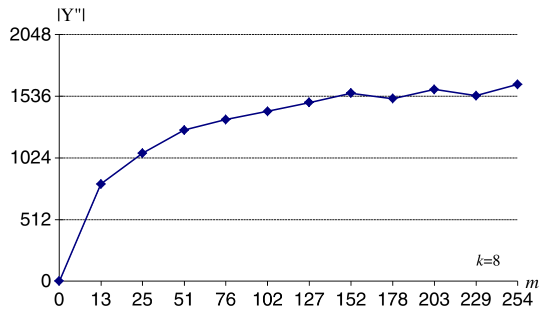


Figure 4. The average variation of the *NoList*.

Complexity of tag initialization algorithm. In the algorithm, there are N tags to be initialized; for each tag, the algorithm produces m pairs of (x, y); and for each y value, the algorithm searches the proximate value of y in the *ylist* using binary search. Here, we let the current number of elements in the *ylist* be $NoList_i$. It is clearly that $NoList_i$ increases gradually while more (x, y) pairs being generated. Figure 4 explores the average variation of the $NoList_i$ under varying m and $k = 8$.
Hence, the complexity of tag initialization algorithm can be estimated as $\sum_{i=1}^{N*m} \log(NoList_i)$, and finally, the size of *ylist* is $NoList_{N*m}$ that equals $|Y''|$. That is, the complexity of tag initialization algorithm will be bounded by $N^*m^* \log(|Y''|) = N^*m^* \log(2^{k''}) = N^*m^*k''$. Accordingly, it is sufficient that k'' is set to 11 for $k = 8$, also refer Figure 4.
Complexity of tag identification algorithm. For each tag identification, the server searches y'' in the *ylist* to map y'' to y and then computes slope s to identify the tag. It takes $\log(|Y''|) = k''$ times in the worse case, a constant-time identification.

4. PERFORMANCE EVALUATIONS

In this section, we use the results of experiments to evaluate the performance of the proposed scheme. The first experiment examines the compression effect of the proposed transformation algorithm on the y value. It adopts $k = 8$ and chooses secret point (a, b) as $(-23, -94)$. Then, it computes all (x, y, y', y'') quartets, where (x, y) is the original point, y' is equal to y or the qualified adjacency of the y, and y'' is the final result, a one-time random number. Figure 5 demonstrates part of raw data. For the rightmost column in Figure 5, it shows the slope, s^*, which is computed by using (x, y') and should be in the range of s plus or minus 0.5. To see the compression effect on y, Figure 6 shows the distributions of (a): (x, y), (b): (x, y'') and (c): (x, y'') in smaller scale, respectively, when $(a, b) = (-23, -94)$. Figure 7 shows the same cases when $(a, b) = (127, -127)$.

From Figures 6 and 7, we can see the original y ranging in $[-2^{15}, 2^{15}]$ can be compressed into the space of $[0, 2^{11}]$, and the result displays a more uniform distribution.

s (tag identity)	x	y	y'	y''	Note: s*
-127	-127	13114	13114	1636	-127
-127	49	-9238	-9238	1102	-127
-127	36	1557	1557	25	-127
-35	-70	1551	1557	25	-35.13
-35	81	-3734	-3719	605	-34.86
1	116	45	46	1463	1.49
1	-9	-80	-80	491	1
51	16	1895	1906	1858	51.28
51	66	4445	4478	91	51.37
125	45	8406	8415	100	125.13
125	-106	-10469	-10508	749	125.47

Figure 5. Part of raw data for $(a, b) = (-23, -94)$ in experiment 1.

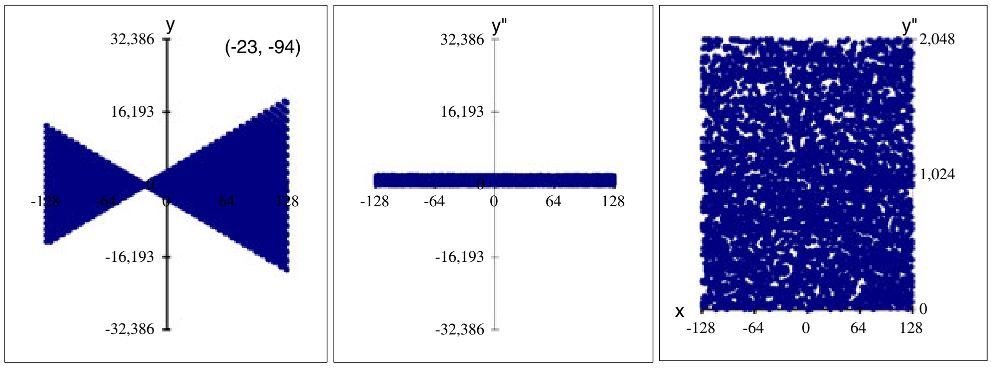
The second experiment explores scalability and corresponding parameter selections. The goal of this experiment is to estimate the size of k'' when $k \geqslant 16$. As we know, k'' can be estimated by the order of Y''. In the experiment, it randomly chooses (a, b), set m as 250 (or 500), then executes tag initialization algorithm five times, and finally computes the average number of elements in Y'', denoted as $|Y''|_{\text{average}}$. Table II shows the results.

According to Table II, we can estimate how many (x, y'') pairs a tag can accommodate under different system scale and different tag storage. Table III shows the results.

5. SECURITY ANALYSIS

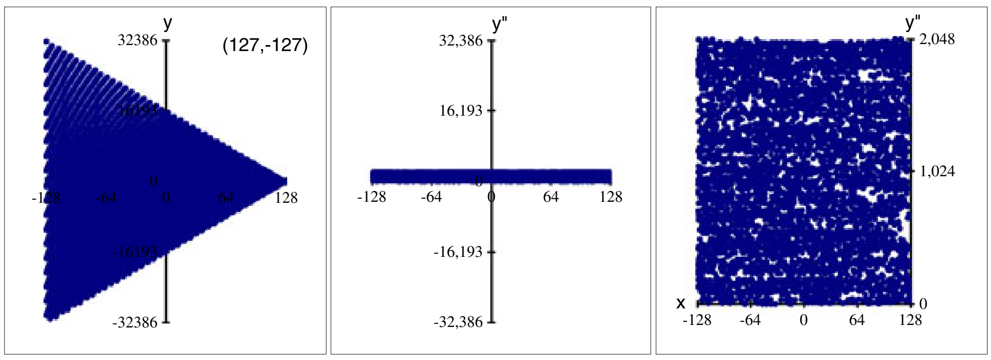
Firstly, we show the security of the proposed scheme in following four dimensions.

Privacy preservation. As shown in the proposed scheme, the proofs, (x, y'') pairs, of a tag are different for each identification. These proofs play as one-time pads. Moreover, (x, y'') can be treated as a random string because x is



(a):The distribution of (*x*,*y*) (b):The distribution of (*x*,*y*") (c):The distribution of (*x*,*y*") in smaller scale

Figure 6. The compression effect on y for $(a, b) = (-23, -94)$ in experiment 1.



(a):The distribution of (*x*,*y*) (b):The distribution of (*x*,*y*") (c):The distribution of (*x*,*y*") in smaller scale

Figure 7. The compression effect on y for $(a, b) = (127, -127)$ in experiment 1.

Table II. The estimation of the size of k''.

k	N	m	$\lvert Y''\rvert_{\text{average}}$	k''
16	65 534	250	470 365 ($\approx 2^{19}$)	19
		500	498 700 ($\approx 2^{19}$)	19
20	1 048 576	250	6 180 011 ($\approx 2^{23}$)	23
		500	10 077 000 ($\approx 2^{24}$)	24
22	4 194 310	250	47 100 995 ($\approx 2^{26}$)	26
		500	108 205 904 ($\approx 2^{27}$)	27

Table III. Parameter selection for different system scale.

k	N	k''	Tag storage (K)	m
16	65 534	19	1	234
			2	468
			3	702
20	1 048 574	24	1	186
			2	372
			3	559
22	4 194 302	27	1	167
			2	334
			3	502

uniformly random whereas y'' is a random string generated by the server. Thus, any two pairs (x_1, y''_1) and (x_2, y''_2) responded by a tag or by two different tags do not have any relationship. Accordingly, the user location cannot be tracked when he or she bears something embedded with such a tag.

Against tag identity guessing. It is a very interesting property of our scheme that an adversary cannot guess the identity of a tag because he or she does not have any extra information about the real y from an eavesdropped y''. Again, any two eavesdropped pairs (x_1, y''_1) and (x_2, y''_2) from a tag are just two random string and thus cannot deduce the identity of the tag.

Against valid $(\mathbf{x}, \mathbf{y''})$ *guessing.* An adversary might select an arbitrary $s \in S$ and consider that it must be the identity of a tag in the system. However, without server's secrecy (a, b), he or she cannot determine the line for the tag. Even if he or she knows (a, b) and thus can computes (x, y), the adversary cannot map y to y'' because the mapping table is preserved by the server. If the adversary uses random guessing, then the successful probability of his or her randomly guessed (x, y'') that can be accepted by the server is bounded by $2^{-(k+k'')}$.

Against $(\mathbf{a}, \mathbf{b})$ *guessing.* Any two lines representing two different tags can compute the interaction point, that is, the point (a, b). However, without any clear y value, an adversary cannot determine any line. It is interesting that even when an adversary wants to exhaustively guess server's (a, b), he or she cannot succeed because none of clear (x, y) can be provided for examining his or her guessing. We refer to this security notion as *unconditionally secure*.

Next, we further analyse the privacy of the proposed scheme in a formal model. We model an adversary $\boldsymbol{A}$ as having the ability to launch either passive or active attacks between server S and arbitrary tag T in an RFID system through the following queries.

- **Execute($\boldsymbol{S}, \boldsymbol{T}, \boldsymbol{i}$)** query
  $\boldsymbol{A}$ eavesdrops on the protocol's flows between the two communicating parties, S and T, in session i of the protocol execution. This query models a passive attack.
- **Send($\boldsymbol{T}$, *message*, $\boldsymbol{i}$)** query
  $\boldsymbol{A}$ impersonates S in session i of the protocol by sending a *message* to tag T. This query models an active attack.
- **Corrupt($\boldsymbol{T}$)** query
  $\boldsymbol{A}$ physically accesses tag T's memory to obtain all stored keys and data. This query also models an active attack.

In an RFID system, *Untraceable Privacy* and *Forward Privacy* are two desirable properties. The former means that given any two uncorrupted tags and historic communication transcripts they were involved in, it is impossible to determine which transcript belongs to which tag. And the latter indicates that even given two corrupted tags and historic communication transcripts they were involved in, it is impossible to determine which transcript belongs to which tag. As follows, we formally define these two privacy notions and prove our RFID identification scheme possesses both of them.

Definition 1. *(Untraceable privacy.)* $\boldsymbol{A}$ *is involved in a game containing two phases as follows.*

Learning phase: $\boldsymbol{A}$ *allows issuing* ***Execute****,* ***Send*** *and* ***Corrupt*** *queries to any tag* T *in the system.*
Challenge phase: $\boldsymbol{A}$ *selects two uncorrupted tags,* T_0 *and* T_1*, to Challenger. Challenger then tosses a random bit* $b = 0$ *or 1 and makes* ***Send***$(T_b$*, message,* $i)$ *to obtain* T_b*'s responding message, idmsg. Finally, Challenger returns* T_b *and idmsg to* $\boldsymbol{A}$*.* $\boldsymbol{A}$ *outputs* b'*.*

In the game, each tag allows being asked $l$times of queries, $l \leqq q$, which may include (i) the possible $(l-1)$ times of **Execute** plus **Send** and the last **Corrupt** query or (ii) the total l times of **Execute** plus **Send** queries. We define an RFID authentication protocol possesses (q, ε)-untraceable privacy if $|\Pr[b' = b] - 1/2| = \varepsilon$ is negligible.

Definition 2. *(Forward privacy.)* $\boldsymbol{A}$ *is involved in a game containing two phases as follows.*

Learning phase: This phase is the same as the one in Definition 1.
Challenge phase: $\boldsymbol{A}$ *sends two uncorrupted tags,* T_0 *and* T_1*, to the Challenger. Challenger then tosses a random bit* b *=0 or 1 and makes* ***Corrupt***(T_b) *to obtain all*

*stored keys and data in T_b's memory. Finally, Challenger returns the corrupted data to **A**. **A** outputs b'.*

The total query times that **A** can make to every tag are also limited as in Definition 1. We define an RFID authentication protocol possesses (q, ε)-forward privacy if $|\Pr[b' = b] - 1/2| = \varepsilon$ is negligible.

Lemma 1. *According to Definition 1, we claim our RFID identification protocol possesses (q, ε)-untraceable privacy.*

Proof. We use Soup's game hopping technique* [32] to prove our claim. Before the proof, we restate our system's parameters. In our system, there are N tags. Each tag's memory is set by m identification messages, *idmsg*s, which each is composed of value x (in length of k bits) and value y (in length of k" bits). Then, we can prove this lemma using the following three games. □

Game0:

Initialization:

This phase adopts the *Tag Initialization Algorithm* described in Section 3. Here, we let $m = q$.

Learning phase:

Adversary **A** *allows calling the following queries* to any T in the system:

(1) Calling **Execute**(S, T, i) and obtaining a transcript {'hello', *idmsg*},
(2) Calling **Send**$(T$, 'hello', $i)$ and obtaining an *idmsg* and
(3) Calling **Corrupt**(T) obtaining all fresh *idmsg*s in T's memory.

Challenge phase:

Adversary **A** selects two uncorrupted tags, T_0 and T_1, to Challenger. Challenger then tosses a random bit $b = 0$ or 1, and calls **Send**$(T_b$, 'hello', $i)$ to obtain an *idmsg*. Finally, Challenger returns T_b and *idmsg* to **A**. **A** outputs b'.

Game1:

Initialization:

For each tag T in the system, set its memory with null value and then do the following steps q times.

(1) Choose a k-bit random string *strx* and a k"-bit random string *stry*, and
(2) Let $str = strx||stry$ and append *str* to T's memory.

Learning phase and *challenge phase* are the same as those in *Game0.*

Game2:

Initialization:

For each tag T in the system, set its memory with null value and then do the following steps q times.

(1) Choose a random string *str* in $(k + k")$ bits, and
(2) Append *str* to T's memory.

Learning phase and *challenge phase* are the same as those in *Game 0.*

In *Game2*, because any identification message answered by a tag in the system is a random string, **A** thus has no extra advantage to decide whether the received *idmsg* is from T_0 or T_1. That is, adversary **A** can only make a random guess for the random bit b. Therefore, $\Pr[b = b'] = 1/2$, that is, Pr[**A** wins *Game2*] is negligible.

In *Game1*, it is obvious that any identification message is composed of two random strings that yield a $(k+k")$-bit random string. Hence, the $(k + k")$-bit identification message in *Game1* is indistinguishable from the identification message in *Game2*. Therefore, Pr[**A** wins *Game1*] = Pr[**A** wins *Game2*].

Finally, an identification message in *Game0* is composed of value x and value y. According to the tag initialization algorithm, x is randomly selected from $[-(2^{k-1} - 1), (2^{k-1} - 1)] - \{a\}$, where a is an integer (see line 8 in the algorithm). When comparing two probability distributions, value x in *Game0* and value *strx* in *Game1*, the difference between them, $\Pr[X = x]$ and $\Pr[X = strx]$, is $|1/2^k - 1/(2^k - 1)| \fallingdotseq 1/2^{2k}$ when k is sufficiently large (If part x is distinguishable, then **A** wins *Game0*). In addition, according to the algorithm, y is randomly selected from $\{0,1\}^{k"}$ (see lines 10, 30, 32, and 24 in the algorithm), and thus is indistinguishable from *stry* in *Game1*. Therefore, Pr[**A** wins *Game0*] = Pr[**A** wins *Game1*] $+1/2^{2k} = 1/2 + 1/2^{2k}$. The value $1/2^{2k}$ is negligible when k is sufficiently large. Therefore, we prove that our RFID identification protocol possesses $(q, 1/2^{2k})$-untraceability privacy.

Lemma 2. *According to Definition 2, we claim our RFID identification protocol possesses (q, ε)-forward privacy.*

Proof. We also use Soup's game hopping technique to prove this claim. The system parameters are the same as the ones in *Lemma 1*. Then, we can prove this lemma using the following three games. □

*Soup's game hopping technique uses a sequence of games, Game 0, Game 1, … and Game n, where Game 0 models the security problem of a new scheme and Game n models a well-known intractable problem or an obviously true statement. In the proof, one should argue the difference of the probabilities of two successive games is equal or negligibly close. As a result, one can prove that the probability of an adversary winning Game 0 is almost same as the probability of winning Game n, and the new problem therefore reduced to the intractable problem or the true statement.)

Game0:

Initialization:

This phase is the one in *Game0* of *Lemma 1*.

Learning phase:

A selects two tags, T_0 and T_1, and makes **Execute** or **Send** queries to both of them. Each tag can be asked at most $(q-1)$ queries.

Challenge phase:

A sends T_0 and T_1 to Challenger. Challenger then tosses a random bit $b = 0$ or 1, and makes **Corrupt**(T_b) to obtain T_b's memory data set, $D = \{idmsg_{v+1}, \ldots, idmsg_q\}$ for $1 \leqq v < q$. Here, we suppose that all the v *idmsg*$_i$, $1 \leqq i \leqq v$, had been queried and erased. In addition, Challenger appends v random strings to D (compensating for the erased ones), which each is composed of two parts, a k-bit random string from $[-(2^{k-1}-1), (2^{k-1}-1)] - \{a\}$ where a is an integer, and a k"-bit random string from $\{0,1\}^{k"}$. Finally, Challenger returns T_b and$D$ to ***A***. ***A*** outputs b'.

Game1:

Initialization:

This phase is the one in *Game1* of *Lemma 1*.

Learning phase:

This phase is the same as the learning phase in *Game0* of *Lemma 2*.

Challenge phase:

This phase is the same as the challenge phase in *Game0* of *Lemma 2*, except that Challenger appends v random strings to D (compensating for the erased ones), which each is composed of a k-bit and a k"-bit random strings.

Game2:

Initialization:

This phase is the same as the one in *Game2* of *Lemma 1*.

Learning phase:

This phase is the same as the learning phase in *Game0* of *Lemma 2*.

Challenge phase:

This phase is the same as the challenge phase in *Game0* of *Lemma 2*, except that Challenger appends v random strings to D (compensating for the erased ones), which each has $(k + k")$ bits.

In *Game2*, because any identification message is a random string, ***A*** thus has no extra advantage to decide whether the received D (which includes q random strings) is from T_0 or T_1. That is, adversary ***A*** can only make a random guess for the random bit b. Therefore, Pr[$b = b'$] $= 1/2$, that is, Pr[***A*** wins *Game2*] is negligible. In *Game1*, it is obvious that any identification message is also a random string. Therefore, Pr[***A*** wins *Game1*] = Pr[***A*** wins *Game2*]. Finally, similar to the same reason described in *Lemma 1*, we have Pr[***A*** wins *Game0*] $\fallingdotseq$ Pr[***A*** wins *Game1*] $+q/2^{2k} = 1/2 + q/2^{2k}$ (value q in the numerator means that if part x in one of the q *idmsg*s is distinguishable, ***A*** wins the game). The value $q/2^{2k}$ is negligible when k is sufficiently large. Therefore, we prove that our RFID identification scheme possesses $(q, q/2^{2k})$-forward privacy.

6. CONCLUSION AND FUTURE WORK

To our best knowledge, our scheme is the first attempt that uses a line on a plane to represent an RFID tag. It assures constant-time tag identification and thus possesses the scalability. Moreover, the random-looking response of a tag is changeable for each identification guards the user location privacy. This paper is just the beginning. Our future work is three aspects. The first is to find another better compression approach for y value to further downsize the space of storing y.

Secondly, the proposed scheme also requires a mechanism of updating (x, y'') pairs in a tag. In our system described in Section 3, there will be no fresh (x, y'') for a tag to respond server/reader's interrogations when m pairs of (x, y'') are used. This will restrict the proposed scheme applied on many applications. A possible solution is that the server reallocates m fresh pairs to the tag via a secure channel.

The third of the future work is to improve the proposed scheme to resist denial-of-service attacks. When suffered intensive malicious interrogations (over m times), a tag in our system will be unavailable anymore. Alomair's solution is to use a counter in a tag, which allows the tag producing many dynamic identities. In addition, the server in Alomair's scheme must store all possible dynamic identities. The solution of RAP series is to launch brute search when the tag had suffered attacks. Although these solutions are not satisfactory, our future work may start from these works.

REFERENCES

1. Lee YK, Batina L, Verbauwhede I. EC-RAC: provably Secure RFID authentication protocol, In *Proceedings in IEEE International Conference on RFID,* IEEE, MGM Grand, Las Vegas, NV USA, 2008; 97–104.
2. Lee YK, Batina L, Verbauwhede I. Untraceable RFID authentication protocols: revision of EC-RAC, In *Proceedings in IEEE International Conference on RFID,* IEEE, MGM Grand, Las Vegas, NV USA, 2009; 178–185.
3. Lee Y, Batina L, Singelée D, Verbauwhede I. Low-cost untraceable authentication protocols for RFID, In *Proceedings in WiSec'10*, New Jersey, USA, March 2010; pp. 55–64.
4. Burmester M, Medeiros BD, Motta R. Anonymous RFID authentication supporting constant-cost

key-lookup against active adversaries. *International Journal of Applied Cryptography* 2008; **1**(2): pp. 79–90.

5. Chen Y, Chou JS, Sun HM. A novel mutual authentication scheme based on quadratic residues for RFID systems. *Computer Networks* 2008; **52**: 2373–2380.
6. Lv C, Li H, Ma J, Zhang Y. Vulnerability analysis of elliptic curve cryptography-based RFID authentication protocols. *Transactions on Emerging Telecommunications Technologies* November 2012; **23**(7): 618–624. DOI: 10.1002/ett.2514.
7. Weis SA, Sarma S, Rivest R, Engels D. Security in Pervasive Computing. In *First International Conference, Boppard, Germany, March 12-14, 2003. Revised Papers*, Vol. 2802, Lecture Notes in Computer Science. Springer: Berlin Heidelberg, 2004; pp 201–212.
8. Ohkubo M, Suzuki K, Kinoshita S. Cryptographic approach to “privacy-friendly” tags, In *Proceedings in RFID Privacy Workshop 2003,* MIT, Orlando, Fla. USA, November 2003; pp. 624–654.
9. Cho JS, Yeo SS, Kim SK. Securing against brute-force attack: a hash-based RFID mutual authentication protocol using a secret value. *Computer Communications* 15 March 2011; **34**(3): 391–397.
10. Avoine G, Dysli E, Oechslin P. Reducing time complexity in RFID systems. *Selected Areas in Cryptography–SAC* 2005; **3897**: 291–306.
11. Lu L, Han J, Hu L, Liu Y, Ni L. Dynamic key-updating: privacy-preserving authentication for RFID systems, In *International Conference on Pervasive Computing and Communications*, San Diego, California, USA, 2007; pp. 13–22.
12. Tsudik G. YA-TRAP: yet another trivial RFID authentication protocol, In *Proceedings in 4th IEEE International Conference Pervasive Computing and Communications (PerCom’06)*, Pisa, Italy, March 2006; pp. 643–646.
13. Le TV, Burnmester M, Medeiros BD. Universally composable and forward secure RFID authentication and authenticated key exchange, In *Proceedings in the Second ACM Symposium on Information, Computer and Communications Security*, Singapore, March 2007; 242–252.
14. Burnmester M, Le TV, Medeiros BD, Tsudik G. Universally composable RFID identification and authentication protocols. *ACM Transactions on Information and Systems Security* 2009; **12**(21). DOI: 10.1145/1513601.1513603.
15. Duc DN, Kim K. Defending RFID authentication protocols against DoS attacks. *Computer Communications* 15 March 2011; **34**(3): 384–390.
16. Alomair B, Clark A, Cuellar J, Poovendran R. Scalable RFID systems: a privacy-preserving protocol with constant-time identification, In *Proceedings in the 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – DSN ’10*, Chicago, Illinois, USA, June 2010. IEEE.
17. Alomair B, Poovendran R. Privacy versus scalability in radio frequency identification systems. *Computer Communications* 15 December 2010; **33**(18): pp. 2155–2163.
18. Lee CF, Chien HY, Laih CS. Server-less RFID authentication and searching protocol with enhanced security. *International Journal of Communication Systems* 2010; **55**(1): 65–79.
19. Sun HM, Ting WC. A Gen2-based RFID authentication protocol for security and privacy. *IEEE Transactions on Mobile Computing* 2009; **8**(8): 1052–1062.
20. Chien HY. SASI: a new ultralightweight RFID authentication protocol providing strong authentication and strong integrity. *IEEE Transactions on Dependable and Security Computing* 2007; **4**(4): 337–340.
21. D’Arco P, Santis AD. On ultralightweight RFID authentication protocols. *IEEE Transactions on Dependable and Security Computing* 2011; **8**(4): 548–563.
22. Tain Y, Chen C, Li J. A new ultralightweight RFID authentication protocol with permutation. *IEEE Communications Letters* 2012; **16**(5): 702–705.
23. Zhang D, Zhou J, Guo M, Cao J. TASA: tag-free activity sensing using RFID tag arrays. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 2011; **22**: 558–570.
24. Juels A. RFID security and privacy: a research survey. *Journal of Selected Areas in Communication* 2006; **24**(2): 381–395.
25. Juels A, Weis SA. Defining strong privacy for RFID. *ACM Transactions on Information and System Security* 2009; **13**(1): pp. 812–815.
26. Molnar D, Wagner D. Privacy and security in library RFID: issues, practices, and architectures, In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS ’04)*, Athens, Greece, October 2004; pp. 210–219.
27. Ryu EK, Takagi T. A hybrid approach for privacy-preserving RFID tags. *Computer Standards & Interfaces* 2009; **31**: 812–815.
28. Lee YK, Sakiyama K, Verbauwhede I. Elliptic-curve-based security processor for RFID. *IEEE Transactions on Computers* 2008; **57**(11): pp. 1514–1527.
29. Gaubatz G, Kaps JP, Ozturk E, Sunar B. State of the art in ultra-low power public key cryptography for wireless sensor networks, In *Proceedings the Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW’05)*, Denver, USA, 2005; pp. 146–150.
30. Gosse F, Standaert FX, Quisquater JJ. FPGA implementation of SQUASH, In *Proceedings in the 29th*

Symposium on Information Theory in the Benelux, Leuven, Belgium, 2008. http://dial.academielouvain.be/vital/access/services/Download/boreal:81797/PDF_01.

31. Feldhofer M, Dominikus S, Wolkerstorfer J. Strong authentication for RFID systems using the AES algorithm. In *Proceedings in the Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), LNCS #3156*, Vol. 3156, Lecture Notes in Computer Science. Springer, 2004; pp. 357–370. 6th International Workshop Cambridge, MA, USA, August 11–13, 2004.
32. Shoup V. Sequences of games: a tool for taming complexity in security proofs, Cryptology ePrint Archive. *Report 2004/332*, 2004. http://eprint.iacr.org/ [Accessed 16 April 2013].