χperbp: a Cloud-based Lightweight Mutual Authentication Protocol

Morteza Adeli[1], Nasour Bagheri[23], Sadegh Sadeghi[4], and Saru Kumari[5]

[1] Mathematics Section, Department of Basic Sciences, Shahid Rajaee Teacher Training University, Tehran, Iran, Postal code: 16788-15811, Tel/fax:+98-21-2297006
m.adeli@sru. ac. ir

[2] Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran 16788-15811, Iran

[3] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran, Nbagheri@sru.ac.ir , na.bagheri@gmail.com

[4] Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran s.sadeghi.khu@gmail.com

[5] Department of Mathematics, Chaudhary Charan Singh University, Meerut 250004, India, Saryusiirohi@gmail.com

Abstract. Alongside the development of cloud computing and Internet of Things(IoT), cloud-based RFID is receiving more attention nowadays. Cloud-based RFID system is specifically developed to providing real-time data that can be fed to the cloud for easy access and instant data interpretation. Security and privacy of constrained devices in these systems is a challenging issue for many applications. To deal with this problem, we propose χperbp, a lightweight authentication protocol based on χper component. χper is a hardware/software friendly component that can be implemented using bit-wise operations. To evaluate the performance efficiency of our proposed scheme, we implement the χperbp scheme on a FPGA module Xilinx Kintex-7 using the hardware description language VHDL. Our security and cost analysis of the proposed protocol shows that the proposed protocol provides desired security against various attacks, in a reasonable cost. Also, formal security evaluation using BAN logic and Scyther tool indicates its security correctness. Besides, we analyse the security of a related protocol which has been recently proposed by Fan *et al.* It is a cloud-based lightweight mutual authentication protocol for RFID devices in an IoT system. Although they have claimed security against active and passive adversaries, however, our detailed security analysis in this paper demonstrates major drawbacks of this protocol. More precisely, the proposed attack disclose the tag's secrets efficiently. Given the tag's secrets, any other attack will be trivial.

Keywords: IoT; Authentication; Security analysis; Desynchronization Attack; Tag Impersonation Attack; Reader Impersonation Attack

1 Introduction

Cloud computing is growing rapidly as the next generation platform for computation, with applications in approximately any area because of its performance, high availability, least cost and many others. On the other hand, in recent years, the use of radio frequency identification (RFID) has increased across a range of different industries such as retail industry, healthcare, transportation and etc. due to its inherent benefits. However, the wide distribution of RFID systems may threaten the security of both businesses and consumers. A cloud-based RFID system, as depicted in Figure 1, is typically composed of three components, namely a tag, a reader and a cloud server. The RFID tag is typically a small device that utilizes low-power radio waves to receive, store, and transmit data to nearby readers and allows users to automatically identify and track inventory and assets. It is comprised of a microchip or integrated circuit (IC) with a small memory to store the object's identity and data, a small antenna and a low power battery(in active tags, passive tags have no power). The RFID reader is a scanner which has more computation and storage resource than the tag and placed in a fixed location to interrogate the tag. The cloud server has considerable resources and in fact, it is the brain of an RFID system which operates as a data processor that manages, controls, and stores the data from the tags and readers.

Fig. 1: A Cloud-Based RFID System

Providing secure communication between these components is regarded as one of the main issues in the RFID systems. In order to satisfy this goal, the authentication protocol of a tag is used in the RFID systems. Authentication protocol is a method to authenticate a remote device like an RFID tag by a reader over an insecure communication channel. Cloud-based authentication is a solution that is quick-to-deploy, easily managed, and support extensive authentication methods. The most common challenge to employ an authentication protocol in such systems is that the tags typically have limited storage and computing resources to support standard cryptographic algorithm such as RSA, ECC, and etc. In addition, an authentication protocol must be secure against various common attacks such as impersonation, replay, de-synchronization and etc. attacks.

The contribution of this paper contains two main folds:

- First, we analyze the Fan *et al.* [17] scheme (called Timestamp-permutation) and show that the proposed scheme is vulnerable to secret disclosure attack. This attack disclose the value of ID_i and its encrypted value $E_1(ID_i)$. Given these values, an adversary can perform other known attacks such as de-synchronization, traceability and so on.
- Second, to address this concern, we proposed an improved lightweight authentication protocol(called χperbp) for IoT applications. We prove the security of the χperbp through formal and informal analysis. At the end, to evaluate the performance efficiency of our proposed scheme, we implement the χperbp scheme on a FPGA module Xilinx Kintex-7 using the hardware description language VHDL and compare the synthesis results with some lightweight schemes.

2 Related works

The evolution of IoT technology drives researchers to design secure and reliable authentication protocol for low-cost RFID systems. However, many challenges arise from using lightweight authentication protocols in RFID systems. For example, some of the proposed schemes are vulnerable to one or more security attacks [2,33,18] and some of them are inefficient in terms of processing time [29,21].

Hoque *et al.* [20] proposed a serverless, forward secure and untraceable authentication protocol for RFID tags. They claimed that their scheme safeguards both tag and reader against almost all major attacks without the intervention of server. However, Deng *et al.* [12] showed that the proposed scheme is vulnerable to de-synchronization attack. They addressed the weakness of the Hoque *et al.* scheme and proposed an improved serverless authentication scheme. In [24], Li *et al.* analyzed the Deng *et al.* scheme and pointed out that this scheme cannot resist location tracking attack, and also its tag searching method is low efficient. Tan *et al.* [32] proposed an authentication protocol that provides comparable protection against known attacks without needing a central authority. However recently, in [34], Wei *et al.* showed that the scheme is vulnerable to denial of service, de-synchronization, and tracking attacks.

In [13], Dhillon *et al.* proposed an authentication scheme for the Internet of Multimedia Things(IoMT) environments. They declared that the scheme is robust and can resist significant security attacks. However, Mahmood *et al.*[25]

showed that it is vulnerable to user masquerading attacks and a stolen verifier attack. Besides, their scheme also violates the anonymity and traceability of a user.

More recently, Fan *et al.* [17] proposed a lightweight cloud-based authentication protocol called Timestamp-permutation for IoT systems. The proposed scheme uses only simple operations such as rotation, permutation, concatenation and a symmetric encryption algorithm. Therefore, it's well suited for using in low-cost applications such as RFID systems. They claimed that the proposed scheme is secure against various known attacks, but in this paper, we show that it is vulnerable to disclosure attack. This attack can disclose the secret information stored in a tag such as the identity ID_i and its encrypted value $E_1(ID_i)$.

3 Timestamp-permutation protocol

In this section, we give briefly description of Timestamp-permutation protocol [17]. This protocol consists of two phases: 1-Initialization 2-Authentication. We represent the notations used in this article in Table 1 and a brief description of Timestamp-permutation scheme in Figure 2. The timestamps in this protocol are based on reader's current time. Before considering this protocol, we need to introduce some definitions.

Definition 1. *Let A, B are two n-bits strings, where*

$$A = a_1 a_2 ... a_n, a_i \in \{0,1\}, i = 1, 2, ..., n$$

$$B = b_1 b_2 ... b_n, b_i \in \{0,1\}, i = 1, 2, ..., n$$

and $C = A \oplus B$ where $C = c_1 c_2 ... c_n$, $c_i \in \{0,1\}$, $i = 1, 2, ..., n$. Moreover let

$$b_{k_1}, b_{k_2}, ..., b_{k_m} = 1$$

$$b_{k_{m+1}}, b_{k_{m+2}}, ..., b_{k_n} = 0$$

where $1 \leq k_1 < k_2 < ... < k_m \leq n$ and $1 \leq k_{m+1} < k_{m+2} < ... < k_n \leq n$. The function $Per(A, B)$ is defined as following:

$$Per(A, B) = c_{k_1} c_{k_2} ... c_{k_m} c_{k_n} c_{k_{n-1}} ... c_{k_{m+2}} c_{k_{m+1}}$$

Definition 2. *Let $wt(B)$ is the Hamming weight of B, where $0 \leq wt(B) \leq n$. The function $Rot(A, B)$ is defined as: A is left routed $wt(B)$ bits.*

Table 1: Notation used in this paper

Notation	Description
$\mathcal{T}_i$	the i-th RFID tag
$\mathcal{C}$	the cloud server
$\mathcal{R}$	the RFID reader
$\mathcal{A}$	the adversary
ID_i	the identity of $\mathcal{T}_i$
$Per(A, B)$	the permutation
$Rot(A, B)$	the rotation
$\theta()$	the obscuring the timestamp
$E_1()\backslash D_1()$	the symmetric encryption\decryption algorithm using a key shared between the readers
$E_2()\backslash D_2()$	the symmetric encryption\decryption algorithm using a key shared between the readers and cloud
$(X)_L$	the left half of X
$(X)_R$	the right half of X
$X \lll i$	rotate X left by i positions
$X \ggg i$	rotate X right by i positions
$\lfloor X \rfloor_i$	assuming $X = x_1x_2 \dots x_n$, then $\lfloor X \rfloor_i = x_{i+1} \dots x_n$
$\lceil X \rceil_i$	assuming $X = x_1x_2 \dots x_n$, then $\lceil X \rceil_i = x_1 \dots x_i$
$\bar{X}^i$	assuming $X = x_1x_2 \dots x_n$, then $\bar{X}^i = x_1 \dots x_{i-1}\bar{x}_i \dots x_n$

3.1 Initialization Phase

We suppose that this phase is conducted in a secure environment. This phase includes the following steps:

1. $\mathcal{T}_i$ stores timestamp T_t, the unique identity ID_i which is assigned by the system and its encryption value $E_1(ID_i)$.
2. $\mathcal{R}$ has the keys of two symmetric encryption algorithms E_1 and E_2.
3. $\mathcal{C}$ stores the encrypted value of each tag's identity and the corresponding timestamps which are followed by a bit "0" or "1". This mark bit is exploited to record which timestamp is more likely to be synchronized with the tag. $\mathcal{C}$ only has the key of the second symmetric encryption algorithm E_2.

3.2 Authentication Phase

1. The reader $\mathcal{R}$ generates a timestamp T_r and sends it to the tag $\mathcal{T}_i$.
2. Upon receiving T_r, the tag computes

$$M_1 = Rot(E_1(ID_i), E_1(ID_i) \oplus T_t)$$

$$M_2 = Per(M_1, E_1(ID_i) \oplus T_r)$$

and sends the messages $\{M_2, \theta(T_t), T_r\}$ to $\mathcal{R}$. Then the reader forwards the messages to $\mathcal{C}$.

3. The cloud $\mathcal{C}$ searches in its database for timestamp T_t which matches $\theta(T_t)$. Then it looks for $E_1(ID_i)$ which matches $Per(M_1, E_1(ID_i) \oplus T_r)$ in the result of the first search. If $E_1(ID_i)$ exists, two states may occur:

   – If the mark bit of T_t is "1", the timestamp marked "0" will be replaced by T_r.

   – If the mark bit of T_t is "0", the last certification may not end normally. T_r will be stored and the previous timestamps will not be deleted.

   Then $\mathcal{C}$ computes $M_3 = E_2(E_1(ID_i)\|T_t\|T_r)$ and sends it to $\mathcal{R}$.

4. $\mathcal{R}$ computes $D_2(E_1(ID_i)\|T_t\|T_r)$ to get $\{E_1(ID_i), T_t, T_r\}$. If it matches with $Per(M_1, E_1(ID_i) \oplus T_r)$ then $\mathcal{R}$ authenticates $\mathcal{C}$. Then the reader decrypts $E_1(ID_i)$ and computes $M_4 = Rot(ID_i, ID_i \oplus T_t)$, $M_5 = Per(M_4, ID_i \oplus T_r)$ and sends $(M_5)_L$ to $\mathcal{T}_i$.

5. Upon receiving, $\mathcal{T}_i$ compares $(M_5)_L$ with $(M_5')_L = (Per(M_4, ID_i \oplus T_r))_L$, if it matches, the tag authenticates the reader and replaces timestamp T_t with T_r. Then it sends $(M_5')_R = (Per(M_4, ID_i \oplus T_r))_R$ to $\mathcal{R}$.

6. If $(M_5')_R$ matches with $(M_5)_R$ then $\mathcal{R}$ authenticates $\mathcal{T}_i$ and sends $M_6 = E_2(E_1(ID_i)\|T_r)$ to $\mathcal{C}$.

7. $\mathcal{C}$ computes $E_2(E_1(ID_i)\|T_r)$ and compares it with M_6. If they matches, $\mathcal{C}$ authenticates $\mathcal{R}$ and updates its database as following:

   – Change the mark bit of timestamp T_r to "1".

   – Delete the timestamps except T_r.

Tag $\mathcal{T}_i$		**Reader** $\mathcal{R}$		**Cloud** $\mathcal{C}$
ID_i,$E_1(ID_i)$,T_t		T_r,E_1,E_2		T_t,T_t',E_2
		Generate T_r		
	$\xleftarrow{Query,T_r}$			
$M_1 = Rot(E_1(ID_i), E_1(ID_i) \oplus T_t)$ $M_2 = Per(M_1, E_1(ID_i) \oplus T_r)$				
	$\xrightarrow[\theta(T_t),T_r]{M_2}$		$\xrightarrow[\theta(T_t),T_r]{M_2}$	
				$M_3 = E_2(E_1(ID_i) \Vert T_t \Vert T_r)$
			$\xleftarrow{M_3}$	
		decrypt M_3 to get $\{E_1(ID_i), T_t, T_r\}$ $M_2 \stackrel{?}{=} Per(M_1, E_1(ID_i) \oplus T_r)$ $M_4 = Rot(ID_i, ID_i \oplus T_t)$ $M_5 = Per(M_4, ID_i \oplus T_r)$		
	$\xleftarrow{(M_5)_L}$			
$m_4' = Rot(ID_i, ID_i \oplus T_t)$ $M_5' = Per(m_4', ID_i \oplus T_r)$ $(M_5')_L \stackrel{?}{=} (M_5)_L$ replace T_t with T_r				
	$\xrightarrow{(M_5')_R}$			
		$(M_5')_R \stackrel{?}{=} (M_5)_R$ $M_6 = E_2(E_1(ID_i) \Vert T_r)$		
			$\xrightarrow{M_6}$	
				$M_6' = E_2(E_1(ID_i) \Vert T_r)$ $M_6' \stackrel{?}{=} M_6$

Fig. 2: Timestamp-permutation protocol

4 Cryptanalysis of Timestamp-permutation protocol

In this section, we analyze the security of the Timestamp-permutation protocol against various attacks. The proposed attacks are based on the observations below:

1. $\mathcal{T}_i$ does not contribute to the session randomness. Hence, as far as it has not updated its timestamp, its response to identical challenge will be the same.
2. On a session of protocol between a legitimate $\mathcal{R}$ and $\mathcal{T}_i$, in Step 1, $\mathcal{R}$ generates a timestamp T_r and sends it to $\mathcal{T}_i$ and in Step 5, $\mathcal{T}_i$ stores it as a new T_t. Hence, a passive adversary $\mathcal{A}$ who monitors the transferred messages of a session over public channel knows the next value of T_t which is used by $\mathcal{T}_i$.

3. Let $A = a_1a_2 \dots a_n$, $B = b_1b_2 \dots b_n$ and $wt(B) = w$. Given $Per(A, B) = x_1x_2 \dots x_n$, then :

$$if\ b_1 = 1 : Per(A, \bar{B}^1) = x_2 \dots x_w x_{w+1} \dots x_n \bar{x}_1$$
$$if\ b_1 = 0 : Per(A, \bar{B}^1) = \bar{x}_n x_1 x_2 \dots x_w x_{w+1} \dots x_{n-1}$$

On the other word:

$$if\ b_1 = 1 : Per(A, \bar{B}^1) = (Per(A, B) \lll 1) \oplus 1$$
$$if\ b_1 = 0 : Per(A, \bar{B}^1) = (Per(A, B) \oplus 1) \ggg 1$$

Following this property, given $Per(A, B)$ and $Per(A, \bar{B}^1)$, one can determine the value of b_1.

4.1 Secret disclosure attack

Following the observation 1, $\mathcal{T}_i$ does not generate any random number. Therefore, the values of T_t and $M_1 = Rot(E_1(ID), E_1(ID) \oplus T_t)$ remains unchanged until $\mathcal{T}_i$ participates in a success session with the reader. According to the observation 2, assume that $\mathcal{A}$ has eavesdropped the last successful session between $\mathcal{T}_i$ and $\mathcal{R}$ and knows the stored value T_t. Then the adversary $\mathcal{A}$ can retrieve $E_1(ID)$ as following:

1. Let $E_1(ID) = e_1e_2...e_n$ and $ID = id_1id_2...id_n$
2. $\mathcal{A}$ impersonates $\mathcal{R}$ by selecting $T_r \in \{0, 1\}^n$ and sending it to $\mathcal{T}_i$.
3. Upon receiving T_r, the tag computes M_1, M_2 and sends the messages $\{M_2, \theta(T_t), T_r\}$ to $\mathcal{R}$.
$$M_1 = Rot(E_1(ID), E_1(ID) \oplus T_t)$$
$$M_2 = Per(M_1, E_1(ID) \oplus T_r)$$
4. $\mathcal{A}$ stores M_2, and sends $\bar{T}_r^1$ to $\mathcal{T}_i$.
5. Upon receiving $\bar{T}_r^1$, the tag computes M_1 and $M_2' = Per(M_1, E_1(ID) \oplus \bar{T}_r^1)$ and returns $\{M_2', \theta(T_t), \bar{T}_r^1\}$.
6. if $M_2' = M_2 \lll 1$ then $e_1 = 1$ otherwise if $M_2' = M_2 \ggg 1$ then $e_1 = 0$.
7. Following this approach, given the value of $\lceil E_1(ID) \rceil_{i-1}$, $\mathcal{A}$ determines e_i as follows:
   (a) $\mathcal{A}$ impersonates the reader by selecting $T_r \in \{0, 1\}^n$ such that $\lceil T_r \rceil_{i-1} \oplus \lceil E_1(ID) \rceil_{i-1} = \{1\}^{i-1}$ and sending it to $\mathcal{T}_i$.
   (b) Upon receiving T_r, the tag computes M_1 and $M_2 = Per(M_1, E_1(ID) \oplus T_r)$ and returns $\{M_2, \theta(T_t), T_r\}$ to the expected $\mathcal{R}$.
   (c) $\mathcal{A}$ stores M_2, and sends $\bar{T}_r^i$ to $\mathcal{T}_i$.
   (d) Upon receiving $\bar{T}_r^i$, the tag computes M_1 and $M_2' = Per(M_1, E_1(ID) \oplus \bar{T}_r^i)$ and returns $\{M_2', \theta(T_t), \bar{T}_r^i\}$ to $\mathcal{R}$, which is indeed $\mathcal{A}$.

(e) if $\lfloor M_2' \rfloor_{i-1} = (\lfloor M_2 \rfloor_{i-1} \lll 1) \oplus 1$ then $e_i = 1$ otherwise if $\lfloor M_2' \rfloor_{i-1} = (\lfloor M_2 \rfloor_{i-1} \oplus 1) \ggg 1$ then $e_i = 0$.

In the following, we describe how an adversary $\mathcal{A}$ can retrieve the whole bits of ID.

1. $\mathcal{A}$ eavesdrops N information sessions of the protocol between $\mathcal{T}_i$ and legitimate $\mathcal{R}$ and blocks the response message $(M_5)_L$. Hence, the T_t is not updated and the adversary $\mathcal{A}$ has $\{T_r^j, T_t, E_1(ID), M_2^j, (M_5^j)_L\}_{j=1}^{j=N}$, where

$$M_5^j = Per(Rot(ID, ID \oplus T_t), ID \oplus T_r^j) \tag{1}$$

2. Given $(M_5)_L$, T_t and T_r, the only unknown value in Equation 1 is the ID's bits. To simplify the index formulation, we remove the indices r, *5* and L for T_r and $(M_5)_L$ respectively. Let

$$T = t_1 t_2 ... t_n \;\; and \;\; \mathbb{T} = \{T_1, ..., T_N\}$$

$$M = m_1 m_2 ... m_{\frac{n}{2}} \;\; and \;\; \mathbb{M} = \{M_1, ..., M_N\}$$

Hence, $\mathcal{A}$ can find the ID's bits as following:

– Suppose that the LSB bit of the $\mathbb{T}_1^1 = \{T_{k_1}, ..., T_{k_{l_1}}\}$ is "1" and the LSB bit of the $\mathbb{T}_1^0 = \{T_{k_{l_1+1}}, ..., T_{k_N}\}$ is "0". We know that the LSB bit of the values $Rot(ID, ID \oplus T_t)$ and ID are fixed, therefore if the LSB bit of $\mathbb{M}_1^1 = \{M_{k_1}, ..., M_{k_{l_1}}\}$ all are the same or the LSB bit $\mathbb{M}_1^0 = \{M_{k_{l_1+1}}, ..., M_{k_N}\}$ are not the same, then we conclude that the $id_1 = 0$, otherwise the $id_1 = 1$.
Given $t_1 \oplus id_1$, there is only two possible bit positions in M that can be occupied due to $t_2 \oplus id_2$. To make the process easier to understand, we modify the elements of the set $\mathbb{M}$ as the following:
  - If $id_1 = 0$ then we shift the elements of the set $\mathbb{M}_1^1 = \{M_{k_1}, ..., M_{k_{l_1}}\}$ one position to the left and put an indicator "x" into their MSB.

(a) Elements of the set $\mathbb{M}_1^1$ (b) Elements of the set $\mathbb{M}_1^0$

Fig. 3: Case $id_1 = 0$

  - Otherwise, if $id_1 = 1$, we do that for elements of the set $\mathbb{M}_1^0 = \{M_{k_{l_1+1}}, ..., M_{k_N}\}$.

(a) Elements of the set $\mathbb{M}_1^1$ (b) Elements of the set $\mathbb{M}_1^0$

Fig. 4: Case $id_1 = 1$

We remain the name of elements of the set $\mathbb{M}$ unchanged after this modification.

– Let assume that the the second bit of the $\mathbb{T}_2^1 = \{T_{k'_1}, ..., T_{k'_{l_2}}\}$ is "1" and the second bit of the $\mathbb{T}_2^0 = \{T_{k'_{l_2+1}}, ..., T_{k'_N}\}$ is "0". Given that second bit of the values $Rot(ID, ID \oplus T_t)$ and ID are fixed, therefore if the LSB bit of $\{M_{k'_1}, ..., M_{k'_{l_2}}\}$ all are the same or the LSB bit $\{M_{k'_{l_2+1}}, ..., M_{k'_N}\}$ are not the same, then we conclude that the $id_2 = 0$, otherwise the $id_2 = 1$. Similarly, if $id_2 = 0$ then we shift the elements of the set $\mathbb{M}_2^1 = \{M_{k'_1}, ..., M_{k'_{l_2}}\}$ one position to the left and put an indicator "x" into their MSB. Otherwise, if $id_2 = 1$ we do that for elements of the set $\mathbb{M}_2^0 = \{M_{k'_{l_2+1}}, ..., M_{k'_N}\}$. We remain the name of elements of the set $\mathbb{M}$ unchanged after this modification.
– We continue this method until the left half bits of the ID are determined(because according to the Timestamp-permutation protocol, we only have the left half of the M_5). To determine the half right bits, we remove the M_js from the set $\mathbb{M}$ which whole of bit positions are occupied with "x" and replace them with new session information $(M_5)_L$s. On average we expect to still have $\frac{N}{2}$ of Ms in the set $\mathbb{M}$ where we are determining the value of id_n.

Given ID and $E_1(ID)$, any other attacks such as tag/reader impersonation attack, traceability attack, de-synchronization attack and so on will be trivial.

Algorithm 1: Disclosure attack algorithm to find the encrypted value $E_1(ID)$

Data: Timestamp T_r
Result: The encrypted value $E_1(ID) = e_1 e_2 ... e_n$

Select T_r ;
Send T_r and $\bar{T}_r^1$ to $\mathcal{T}_i$ and store its response M_2 and M'_2 respectively;
if $(M'_2 = (M_2 \lll 1) \oplus 1)$ **then**
  $e_1 = 1$;
else
  $e_1 = 0$;
for *i=2 to 128* **do**
  Select $T_r \in \{0,1\}^n$ such that $\lceil T_r \rceil_{i-1} \oplus \lceil (E_1(ID) \rceil_{i-1} = \{1\}^{i-1}$ and send it and $\bar{T}_r{}^i$ to $\mathcal{T}_i$;
  if $(\lfloor M'_2 \rfloor_{i-1} = (\lfloor M_2 \rfloor_{i-1} \lll 1) \oplus 1)$ **then**
    $e_i = 1$;
  else
    $e_i = 0$;

Algorithm 2: Disclosure attack algorithm to find the identity ID

Data: Timestamp $T_r, (M_5)_L$

Result: The identity $ID = id_1 id_2 ... id_n$

```
Eavesdrop $\{(T_r^j, (M_5^j)_L)\}_{j=1}^{j=N}$ ;
for i=1 to n/2 do
    Construct the sets $(\mathbb{T}_i^1, \mathbb{M}_i^1)$,$(\mathbb{T}_i^0, \mathbb{M}_i^0)$;
    if (the LSB bits of $\mathbb{M}_i^1$ all are the same or the LSB bits of $\mathbb{M}_i^0$ are not the same) then
        $id_i = 0$;
        shift the elements of the set $\mathbb{M}_i^1$ one position to the left ;
    else
        $id_i = 1$ ;
        shift the elements of the set $\mathbb{M}_i^0$ one position to the left ;
```

5 Improved protocol

The main drawback of the Fan *et al.* [17] scheme which leads to the disclosure attack, is the lack of a nonlinear function. Hence, it can not provide enough confusion, as a criteria to design a secure primitive. Following the Shannon's idea, any secure primitive should provide confusion and diffusion [3]. However, the proposed $Per(Rot(.))$ function only provides diffusion property. To add the confusion property into the previous scheme, we use a nonlinear function χ which is used in the Keccak [4] algorithm in our improved scheme. Keccak was standardized as SHA-3 hash function by NIST. χ function is an adjustable permutation for any odd value and we use a variant with 3 bits input-output. Using this nonlinear component, we introduce $\chi per(A, B) : \{0,1\}^{3w} \times \{0,1\}^{3w} \rightarrow \{0,1\}^{3w}$, as depicted in Figure 6, where each variable consist of 3 words and w denotes a word length. $\chi per(.)$ is used to design a general function called $\chi per^z(.)$. Algorithm 3 describes $\chi per^z(.)$, which includes z call to $\chi per(.)$. The variables z and w provides trade off between efficiency and security. Our recommendations for w and z are $w = 32$ and $z \geq 16$. In A, the security of $\chi per^z(.)$ function has been investigated against several known attacks.

Given $\chi per^z(.)$, we redesign the protocol of Fan *et al.* and call it χperbp, stands for χper based protocol.

5.1 Initialization Phase of χperbp

This phase of the improved protocol includes the following steps:

1. $\mathcal{T}_i$ stores the timestamp T_t, the unique identity ID_i which is assigned by the system and its secret key value K_i, shared by $\mathcal{C}$. We also assume that each tag is equipped with a $\chi per^z(.)$ function.

2. $\mathcal{R}$ has its identifier RID and its key K_r, shared with $\mathcal{C}$. The reader is equipped with $\chi per^z(.)$, a 48-bit $PRNG(.)$ and a secure hash function $H(.)$, e.g., PHOTON [19].

3. $\mathcal{C}$ stores the key and the identifier of $\mathcal{R}$ and $\mathcal{T}_i$.

5.2 Authentication Phase of χperbp

The authentication phase of χperbp is defined as below:

1. The reader $\mathcal{R}$ generates a random number R_r and sends it to the tag $\mathcal{T}_i$.

2. Upon receiving R_r, the tag computes two values $R_t = \chi per^z((T_t \| R_r), K_i)$ and $M_t = \chi per^z(ID_i \oplus (R_r \| (R_t)_R), K_i)$ and then sends the messages $\{M_t, (R_t)_R\}$ to $\mathcal{R}$. Afterwards, it replaces the value T_t with $(R_t)_L$ and stores it in its local memory.

3. The reader $\mathcal{R}$ extracts its timestamp T_r and computes $MAC_r = H(M_t \| (R_t)_R \| R_r \| K_r \| T_r \| RID)$. Then it sends $\{M_t, (R_t)_R, R_r, T_r, MAC_r\}$ to $\mathcal{C}$.

4. The cloud $\mathcal{C}$ checks timestamp T_r to make sure it's in a reasonable delay time and searches in its database for the RID, based on the received MAC_r to authenticate the reader $\mathcal{R}$. Then, it searches in its database for a record of a tag which is matched to M_t to authenticate the tag $\mathcal{T}_i$. Next, $\mathcal{C}$ extracts its timestamp T_c, computes $M_c = \chi per^z(ID_i, K_i \oplus (T_c \| (R_t)_R))$, $DI_i = ID_i \oplus RID \oplus \chi per^z(T_c \| T_r, K_r)$ and $MAC_c = H(M_c \| M_t \| R_r \| (R_t)_R \| RID \| ID_i \| T_c)$ and sends $\{MAC_c, DI_i, M_c, T_c\}$ to $\mathcal{R}$.

5. $\mathcal{R}$ extracts the value ID_i from DI_i and verifies the received T_c and MAC_c to authenticate $\mathcal{C}$ and $\mathcal{T}_i$. Then, it computes $M_r = \chi per^z(M_t \oplus M_c, ID_i)$ and sends $\{M_c, M_r, T_c\}$ to $\mathcal{T}_i$.

6. Once received the message, $\mathcal{T}_i$ verifies whether $M_c \stackrel{?}{=} \chi per^z(ID_i, K_i \oplus (T_c \| (R_t)_R))$ to authenticate $\mathcal{C}$. Then it authenticates the reader $\mathcal{R}$ using M_r.

Tag $\mathcal{T}_i$		**Reader** $\mathcal{R}$		**Cloud** $\mathcal{C}$
ID_i,K_i,T_t		RID,K_r		ID_i,K_i,RID,K_r
		Generate R_r		
	$\xleftarrow{Query,R_r}$			
$R_t = \chi per^z((T_t\|R_r),K_i)$ $M_t = \chi per^z(ID_i \oplus (R_r\|(R_t)_R), K_i)$ Replace T_t with $(R_t)_L$				
	$\xrightarrow{M_t,(R_t)_R}$	Extract T_r $MAC_r = H(M_t\|(R_t)_R\|R_r \|K_r\|T_r\|RID)$		
			$\xrightarrow[M_t,R_r,(R_t)_R]{MAC_r,T_r}$	Verify T_r Authenticate $\mathcal{R}$ based on MAC_r Authenticate $\mathcal{T}_i$ based on M_t Extract T_c $M_c = \chi per^z(ID_i, K_i \oplus (T_c\|(R_t)_R))$ $MAC_c = H(M_c\|M_t\|R_r\| (R_t)_R\|RID\|ID_i\|T_c)$ $DI_i = ID_i \oplus RID \oplus \chi per^z(T_c\|T_r, K_r)$
		Verify T_c and extract ID_i Verify MAC_c, authenticate $\mathcal{C}$ and $\mathcal{T}_i$ $M_r = \chi per^z(M_t \oplus M_c, ID_i)$	$\xleftarrow[T_c,M_c]{MAC_c,DI_i}$	
	$\xleftarrow[M_r]{T_c,M_c}$			
Verify M_c and M_r Authenticate $\mathcal{C}$ based on M_c Authenticate $\mathcal{T}_i$ based on M_r				

Fig. 5: Illustration of the authentication phase of χperbp

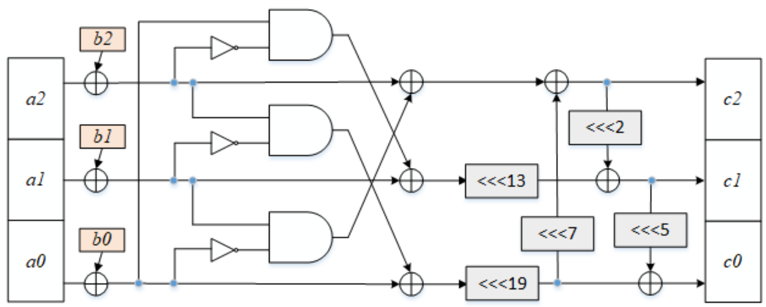

Fig. 6: $C = \chi per(A, B)$

Algorithm 3: $\chi per^z(A, B)$ based on $\chi per(A, B)$

Data: $A = a_0 \| a_1 \| a_2$ and $B = b_0 \| b_1 \| b_2$

Result: $\chi per^z(A, B)$

for *i=0 to 2* **do**

  $x_{i,0} = a_i$ and $y_{i,0} = b_i$;

for *i=0 to z-1* **do**

  $X_i = x_{0,i} \| x_{1,i} \| x_{2,i}$, $Y_i = y_{0,i} \| y_{1,i} \| y_{2,i}$;

  $X_{i+1} \leftarrow \chi per(X_i, Y_i)$;

  $Y_{i+1} \leftarrow (Y_i \lll 95) \oplus 0x243f6a8823ac08e1cb7a0379$;

Return X_z.

6 Security Analysis of the χperbp Protocol

In this section, firstly we analyze the informal security of our proposed scheme against the attacks proposed in this paper and then, using formal security analysis under the broadly-accepted Burrows-Abadi-Needham (BAN) logic and an automated security analysis tool Scyther, we show that the χperbp protocol is secure against various known attacks. At the end of this section, we show the security comparison of the improved scheme with some relevant schemes in Table 4.

6.1 Informal security analysis

Replay attack In this attack, an adversary tries to eavesdrop some communication information and resend them to the tag, reader or the server in another time. In the improved scheme χperbp, we use two random numbers R_t, R_r along with two timestamps T_r, T_c for each session to preventing the replay attack.

Impersonation attack Assume an adversary tries to impersonate himself/herself as a legal tag to cloud server. He/she is not able to produce a valid request message M_t because the adversary needs to know the user's identity ID_i and shared password key K_i between the tag and the cloud. Also the adversary cannot impersonate himself/herself as a legal cloud server because he/she is not able to produce M_c. Therefore the χperbp scheme is secure against impersonate attack.

Traceability and anonymity In χperbp scheme, all transferred messages between three parties tag, reader and the cloud server include at least one of the random numbers R_t, R_r or timestamps T_r, T_c which are updated in each session. Therefore an adversary cannot trace a particular tag since tag's responses to a fixed query is always different at the valid sessions.

Secret disclosure attack The weakness of the Fan *et al.* scheme that deal to disclosure attack is the lack of a nonlinear function. In χperbp scheme, we use $\chi per^z(.)$ function which satisfies the confusion property significantly. Therefore an adversary is not able to carry out disclosure attack same as described in section 4.

De-synchronization attack In χperbp scheme, we use two timestamps T_r and T_c to synchronize the reader and cloud. The T_r value concatenates with $\{M_t, (R_t)_R, R_r, K_r, RID\}$ and the T_c value concatenates with $\{M_c, M_t, R_r, (R_t)_R, RID, ID_i\}$, then both of them are hashed. Therefore the attacker can not change the values T_r and T_c, because he/she must compute the MAC_r and MAC_c, but he/she doesn't know the values of the ID_i, RID and K_r.

A man-in-the-middle attack The communications between the reader and the cloud are hashed, therefore if the attacker intercepts the messages $\{T_r, M_t, R_r, (R_t)_R\}$ or $\{DI_i, T_c, M_c\}$, he/she cannot compute the MAC_r and MAC_c because he/she doesn't know the values of the ID_i, RID and K_r. Also, the tag verifies the received messages with χper function, so the χperbp is secure against man-in-the-middle attack.

6.2 Formal security analysis using BAN logic

To correct evaluate about the χperbp scheme, we use BAN Logic [8] proposed by Burrows, Abadi and Needham. The BAN logic provides a formal method for reasoning about the beliefs of principals in cryptographic protocols. From a practical viewpoint, the analysis of a protocol is performed as follows:

- Transform message into idealized logical formula
- State assumptions about original message
- Make annotated idealized protocols for each protocol statement with assertions

– Apply logical rules to assumptions and assertions
– Deduce beliefs held at the end of protocol

We present the notations and rules used in BAN logic proof in Table 2 and Table 3. The steps of our formal security analysis are as follows:

Table 2: BAN logic notations

Notation	Description
$A\| \equiv X$	A believes X
$A \triangleleft X$	A receives X
$A\| \sim X$	A sends X
$\#(X)$	X is fresh
$A \overset{k}{\leftrightarrow} B$	A and B have a shared secret k
$\{X\}_k$	X is encrypted by the secret key k
$A\| \Rightarrow X$	A regulates X
$< X >_k$	X is exclusive OR-ed with k
$H(X)$	Hash of X

Table 3: BAN logic rules

Rule	Description
$R1 : \frac{A\|\equiv A \overset{k}{\leftrightarrow} B, A \triangleleft \{X\}_k}{A\|\equiv B\|\sim X}$	A believes that B has sent X to him/her when A believes that he/she shared key k with B and received the encrypted message $\{X\}_k$
$R2 : \frac{A\|\equiv B\|\sim H(X), A \triangleleft X}{A\|\equiv B\|\sim X}$	A believes that B has sent X to him/her when A believes that B has sent hashed value $H(X)$
$R3 : \frac{A\|\equiv B\|\sim (X,Y)}{A\|\equiv B\|\sim X}$	A believes that X has been sent by B when he/she believes B has sent (X,Y)
$R4 : \frac{A\|\equiv \#(X)}{A\|\equiv \#(X,Y)}$	A believes that if X is fresh then (X,Y) is fresh

– **Step 1. All transmitted messages of the protocol**: In this step, we list all transmitted messages of the χperbp scheme as bellow:

$M1 : \mathcal{R} \rightarrow \mathcal{T}_i : R_r, Query.$
$M2 : \mathcal{T}_i \rightarrow \mathcal{R} : (R_t)_R = \chi per^z((T_t \| R_r), K_i), M_t = \chi per^z(ID_i \oplus (R_r \| (R_t)_R), K_i).$
$M3 : \mathcal{R} \rightarrow \mathcal{C} : MAC_r = H(M_t \| (R_t)_R \| R_r \| K_r \| T_r \| RID), M_t, R_r, (R_t)_R, T_r.$
$M4 : \mathcal{C} \rightarrow \mathcal{R} : M_c = \chi per^z(ID_i, K_i \oplus (T_c \| (R_t)_R)), MAC_c = H(M_c \| M_t \| R_r$
$\| (R_t)_R \| RID \| ID_i \| T_c), DI_i = ID_i \oplus RID \oplus \chi per^z(T_c \| T_r, K_r), T_c.$
$M5 : \mathcal{R} \rightarrow \mathcal{T}_i : M_c, M_r = \chi per^z(M_t \oplus M_c, ID_i), T_c.$

– **Step 2. Idealizing the messages of the protocol**: In this step, using the BAN logic notations, we express idealized form of the messages in the previous step.

  $IM1 : \mathcal{T}_i \triangleleft R_r, Query.$
  $IM2 : \mathcal{R} \triangleleft \{(R_t)_R, M_t\}_{K_i}.$
  $IM3 : \mathcal{C} \triangleleft H(M_t, (R_t)_R, R_r, K_r, T_r, RID), \{M_t, (R_t)_R\}_{K_i}, T_r, R_r.$
  $IM4 : \mathcal{R} \triangleleft \{M_c\}_{K_i}, \{DI_i\}_{K_r}, H(M_c, M_t, R_r, (R_t)_R, RID, T_c, ID_i), T_c.$
  $IM5 : \mathcal{T}_i \triangleleft \{M_c\}_{K_i}, \{M_r\}_{ID_i}, T_c.$

– **Step 3. Explicit assumptions**: The explicit assumptions of the χperbp scheme are listed as following:

  $A1 : \mathcal{R}| \equiv \#(R_r).$
  $A2 : \mathcal{T}_i| \equiv \#(R_t).$
  $A3 : \mathcal{R}| \equiv \#(T_r).$
  $A4 : \mathcal{C}| \equiv \#(T_c).$
  $A5 : \mathcal{T}_i| \equiv \mathcal{T}_i \xleftrightarrow{K_i} \mathcal{C}.$
  $A6 : \mathcal{C}| \equiv \mathcal{C} \xleftrightarrow{K_i} \mathcal{T}_i.$
  $A7 : \mathcal{R}| \equiv \mathcal{R} \xleftrightarrow{K_r} \mathcal{C}.$
  $A8 : \mathcal{C}| \equiv \mathcal{C} \xleftrightarrow{K_r} \mathcal{R}.$

– **Step 4. Security goals of the protocol**: The security goals that the χperbp scheme must meet are as follows:

  $G1 : \mathcal{C}| \equiv \mathcal{T}_i| \sim ID_i.$
  $G2 : \mathcal{C}| \equiv \mathcal{R}| \sim RID.$
  $G3 : \mathcal{R}| \equiv \mathcal{C}| \sim RID.$
  $G4 : \mathcal{R}| \equiv \mathcal{C}| \sim ID_i.$
  $G5 : \mathcal{T}_i| \equiv \mathcal{C}| \sim ID_i.$
  $G6 : \mathcal{T}_i| \equiv \mathcal{R}| \sim ID_i.$

– **Step 5. Proving the security goals of the protocol**:

  *Result*1: From the $R1$, $A5$, $A3$ and $IM3, IM2$, the goal $G1$ is proved.
  *Result*2: According to the $R2$, $A8$ and $IM3$, the goal $G2$ is proved.
  *Result*3: Given the $R2$, $A1$, $A7$ and $IM4$, the goal $G3$ is proved.
  *Result*4: According to the $IM4$, $R1$, $R2$, $R3$, $A4$ and $A7$, the goal $G4$ is proved.
  *Result*5: Given the $IM5$, $A5$ and $R1$, the goal $G5$ is proved.
  *Result*6: Given the $R1$, $R4$, $A2$, $A4$, $A7$ and $IM5$, the goal $G6$ is proved.

6.3 Automated verification through Scyther tool

We use Scyther tool [9] to verify the correctness and security of the χperbp scheme. Scyther is an automated security protocol analysis tool under the perfect cryptography assumption, in which it is assumed that the adversary learns nothing from the encrypted or hashed data. We describe the specification of a security protocol by a set of roles such as tag's role, reader's role and server's role. Roles are defined by a sequence of events such as sending or receiving of terms. Scyther's input language is SPDL, therefore we write χperbp scheme in SPDL language as depicted in B. To learn more about Scyther tool and SPDL language, we refer the reader to [10,9]. Report of Scyther tool, as depicted in Figure 7, shows that the χperbp scheme is secure against known attacks.

Table 4: Security comparison

	SDA	ImA	DeA	RA	TA	FBSA	MIMA	AA
Ref [1]	✓	✓	✓	✓	×	×	✓	✓
Ref [27]	✓	✓	✓	✓	×	×	×	×
Ref [15]	×	×	✓	✓	×	✓	✓	✓
Ref [16]	×	×	×	✓	✓	✓	✓	×
Ref [17]	×	×	×	×	×	×	×	×
χperbp	✓	✓	✓	✓	✓	✓	✓	✓

SDA : secret disclosure attack
ImA : impersonation attack
DeA : de-synchronization attack
RA : replay attack
TA : traceability attack
FBSA : forward-backward security attack
MIMA : man-in-middle attack
AA : anonymity attack

Improved	Tag	Improved,Tag1	Secret IDi	Ok	No attacks within bounds.
		Improved,Tag2	Secret Ki	Ok	No attacks within bounds.
		Improved,Tag3	Niagree	Ok	No attacks within bounds.
		Improved,Tag4	Nisynch	Ok	No attacks within bounds.
		Improved,Tag5	Alive	Ok	No attacks within bounds.
		Improved,Tag6	Weakagree	Ok	No attacks within bounds.
	Reader	Improved,Reader1	Secret IDi	Ok	No attacks within bounds.
		Improved,Reader2	Secret Kr	Ok	No attacks within bounds.
		Improved,Reader3	Secret RID	Ok	No attacks within bounds.
		Improved,Reader4	Niagree	Ok	No attacks within bounds.
		Improved,Reader5	Nisynch	Ok	No attacks within bounds.
		Improved,Reader6	Alive	Ok	No attacks within bounds.
		Improved,Reader7	Weakagree	Ok	No attacks within bounds.
	CloudServer	Improved,CloudServer1	Secret IDi	Ok	No attacks within bounds.
		Improved,CloudServer2	Secret Ki	Ok	No attacks within bounds.
		Improved,CloudServer3	Secret Kr	Ok	No attacks within bounds.
		Improved,CloudServer4	Secret RID	Ok	No attacks within bounds.
		Improved,CloudServer5	Niagree	Ok	No attacks within bounds.

Done.

Fig. 7: Scyther tool results

6.4 Performance analysis

The χperbp scheme uses two main security functions: the $\chi per^z(.)$ function and a hash function. In the tag side, which has limited resources, the $\chi per^z(.)$ function only need to be implemented. We implement the $\chi per^z(.)$ function on the FPGA module Xilinx Kintex-7 using the hardware description language VHDL. Synthesis and simulation of the HDL code is executed using Vivado v2018.3. As mentioned in section 5, security and performance of the $\chi per^z(.)$ function depends on the two parameters w and z. We recommend $w = 32$ and $z \geq 16$, therefore, based on these values, we calculate the throughput, *tp*, and the throughput-area ratio, *tp-area* of the $\chi per^z(.)$ algorithm by the following formula:

$$tp = \frac{Block\ size}{Cycles\ per\ block} \times Frequency(Mhz)$$

$$tp\text{-}area = \frac{tp}{Slice\,LUTs}$$

The throughput and implementation cost comparison of the $\chi per^z(.)$ function with some lightweight encryption functions which are used in the lightweight authentication schemes is shown in Table 5. Furthermore, we also implement the $Per(Rot(.))$ function which acts as a major function in the Timestamp-permutation protocol.

As shown in Table 5, the device utilization of the simulation after synthesis of the $\chi per^z(.)$ is 460 look-up-tables (LUTs) and its clock rate (frequency) is 680(Mhz). Moreover, $\chi per^z(.)$ function has the highest $tp/area$ which shows that it is more lightweight than the others.

An RTL schematic of the $\chi per(.)$ function is depicted in Figure 8. In this figure, the $\chi per(.)$ function is represented in terms of logic gates such as AND, NAND, and OR. In this diagram, 96-bit plaintext ($A = a1\|a2\|a3$) and 96-bit secret key ($B = b1\|b2\|b3$) are inputs, and 96-bit C is the output.

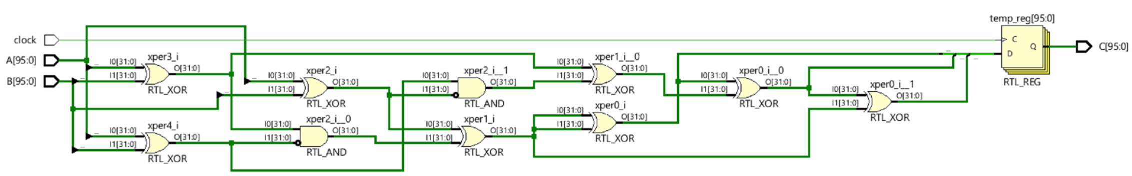

Fig. 8: Logic diagram of the synthesized $\chi per(.)$ function

Table 5: Throughput and implementation cost for various functions [14][22]

Function	Area (LUT)	Frequency (Mhz)	Throughput (Mbps)	Throughput/Area (Mbps/LUT)
SIMON-96	435	564	1041	2.39
SPECK-96	452	473	1622	3.59
PRESENT-80	311	542	1084	3.49
Blake	251	211	477	1.90
Keccak	393	159	864	2.19
Per(Rot(.))-80	904	244	81	0.08
$\chi per^z(.)$-96	460	680	10880	23.65

7 Conclusion

In this paper, we analyzed the Timestamp-permutation protocol proposed by Fan *et al.* for IoT applications and showed that their scheme is vulnerable to disclosure attack. This attack can disclose all the secret information stored on a tag such as the identity of the tag ID_i and its encryption value $E_1(ID_i)$. This attack is practical because it requires at most 128 session information. This values can be used to other attacks such as impersonate attack, de-synchronization attack, replay attack and etc. The permutation function used in the Timestamp-permutation scheme has not good confusion property and this weakness lead to the disclosure attack. To address this vulnerability, we use a nonlinear function called $\chi per^z(.)$ and redesign the Timestamp-permutation scheme. We implement the $\chi per^z(.)$ function on a Xilinx Kintex-7 FPGA using VHDL language and compare the implementation cost with some lightweight encryption functions. The security and performance comparison results of the χperbp show that this protocol is well suited for resource-constrained environments such as RFID tags and sensor nodes.

As a limitation of χperbp, we should mention that to find the tag through the authentication phase, the server should search whole database. Although, the server could have enough computation resources, however, it is a shortcoming in any application for which scalability is important. Hence, as a future work, we suggest to improve this feature of the protocol. In addition, χper is a new primitive which can be used in any other protocol independent of χperbp. In this paper, we have shown its security against various attack, but we encourage other researchers to investigate its security independently.

References

1. S. Abughazalah, K. Markantonakis, and K. Mayes. Secure improved cloud-based rfid authentication protocol. In J. Garcia-Alfaro, J. Herrera-Joancomartí, E. Lupu, J. Posegga, A. Aldini, F. Martinelli, and N. Suri, editors, *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*, pages 147–164, Cham, 2015. Springer International Publishing.
2. M. Adeli and N. Bagheri. Mdsbsp: a search protocol based on mds codes for rfid-based internet of vehicle. *The Journal of Supercomputing*, 2020.
3. D. R. Anderson. Information theory and entropy. *Model based inference in the life sciences: A primer on evidence*, pages 51–82, 2008.
4. G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Keccak. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 313–314. Springer, 2013.
5. E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 12–23. Springer, 1999.
6. E. Biham and A. Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72, 1991.

7. A. Bogdanov and V. Rijmen. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Designs, codes and cryptography*, 70(3):369–383, 2014.
8. M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.
9. C. Cremers. Cispa. `https://people.cispa.io/cas.cremers/publications/index.html`.
10. C. Cremers, S. Mauw, and A. Samarin. *Operational Semantics and Verification of Security Protocols*. Information Security and Cryptography. Springer-Verlag Berlin Heidelberg, 2012.
11. T. Cui, K. Jia, K. Fu, S. Chen, and M. Wang. New Automatic Search Tool for Impossible Differentials and Zero-Correlation Linear Approximations. *IACR Cryptol. ePrint Arch.*, 2016:689, 2016.
12. M. Deng, W. Yang, and W. Zhu. Weakness in a serverless authentication protocol for radio frequency identification. In W. Wang, editor, *Mechatronics and Automatic Control Systems*, pages 1055–1061, Cham, 2014. Springer International Publishing.
13. P. K. Dhillon and S. Kalra. Secure multi-factor remote user authentication scheme for internet of things environments. *International Journal of Communication Systems*, 30(16):e3323, 2017.
14. W. Diehl, F. Farahmand, P. Yalla, J. Kaps, and K. Gaj. Comparison of hardware and software implementations of selected lightweight block ciphers. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4, 2017.
15. K. Fan, W. Jiang, H. Li, and Y. Yang. Lightweight rfid protocol for medical privacy protection in iot. *IEEE Transactions on Industrial Informatics*, 14(4):1656–1665, 2018.
16. K. Fan, J. Kang, S. Zhu, H. Li, and Y. Yang. Permutation matrix encryption based ultralightweight secure RFID scheme in internet of vehicles. *Sensors*, 19(1), 2019.
17. K. Fan, Q. Luo, K. Zhang, and Y. Yang. Cloud-based lightweight secure rfid mutual authentication protocol in iot. *Information Sciences*, 527:329 – 340, 2020.
18. L. Gao, L. Zhang, F. Lin, and M. Ma. Secure rfid authentication schemes based on security analysis and improvements of the usi protocol. *IEEE Access*, 7:8376–8384, 2019.
19. J. Guo, T. Peyrin, and A. Poschmann. The PHOTON family of lightweight hash functions. In P. Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011.
20. M. E. Hoque, F. Rahman, S. I. Ahamed, and J. H. Park. Enhancing privacy and security of rfid system with serverless authentication and search protocols in pervasive environments. *Wireless Personal Communications*, 55:65 – 79, 2010.
21. M. M. Islam, M. S. Hossain, M. K. Hasan, M. Shahjalal, and Y. M. Jang. Design and implementation of high-performance ecc processor with unified point addition on twisted edwards curve. *Sensors*, 20(18):5148, Sep 2020.
22. B. Jungk and J. Apfelbeck. Area-efficient fpga implementations of the sha-3 finalists. In *2011 International Conference on Reconfigurable Computing and FPGAs*, pages 235–241, 2011.
23. L. Knudsen. Deal-a 128-bit block cipher. *complexity*, 258(2):216, 1998.
24. J. Li, Z. Zhou, and P. Wang. Server-less lightweight authentication protocol for rfid system. In X. Sun, H.-C. Chao, X. You, and E. Bertino, editors, *Cloud Computing and Security*, pages 305–314, Cham, 2017. Springer International Publishing.

25. K. Mahmood, W. Akram, A. Shafiq, I. Altaf, M. A. Lodhi, and S. H. Islam. An enhanced and provably secure multi-factor authentication scheme for internet-of-multimedia-things environments. *Computers and Electrical Engineering*, 88:106888, 2020.
26. M. Matsui. Linear cryptanalysis method for des cipher. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 386–397. Springer, 1993.
27. M. Mohammedi, M. Omar, and A. Bouabdallah. Secure and lightweight remote patient authentication scheme with biometric inputs for mobile healthcare environments. *Journal of Ambient Intelligence and Humanized Computing*, 9(5):1527–1539, Oct 2018.
28. N. Mouha, Q. Wang, D. Gu, and B. Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *International Conference on Information Security and Cryptology*, pages 57–76. Springer, 2011.
29. M. Nikooghadam and H. Amintoosi. Perfect forward secrecy via an ecc-based authentication scheme for sip in voip. *The Journal of Supercomputing*, 76:3086 – 3104, 2020.
30. Y. Sasaki and Y. Todo. New impossible differential search tool from design and cryptanalysis aspects. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 185–215. Springer, 2017.
31. S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song. Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES (L) and other bit-oriented block ciphers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 158–178. Springer, 2014.
32. C. C. Tan, B. Sheng, and Q. Li. Secure and serverless rfid authentication and search protocols. *IEEE Transactions on Wireless Communications*, 7(4):1400–1407, 2008.
33. E. Vahedi, R. K. Ward, and I. F. Blake. Security analysis and complexity comparison of some recent lightweight rfid protocols. In Á. Herrero and E. Corchado, editors, *Computational Intelligence in Security for Information Systems*, pages 92–99, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
34. C.-H. Wei, C.-Y. Yang, and M.-S. Hwang. Cryptanalysis of the serverless rfid authentication and search protocols. In F. Xhafa, S. Patnaik, and M. Tavana, editors, *Advances in Intelligent, Interactive Systems and Applications*, pages 842–846, Cham, 2019. Springer International Publishing.
35. S. Wu and M. Wang. Security evaluation against differential cryptanalysis for block cipher structures. *IACR Cryptology ePrint Archive*, 2011:551, 2011.

A Security Analysis of χper function

In this section, we present the results of our security analysis of χper against differential [6], linear [26], impossible differential [23,5] and zero-correlation [7] attacks. To investigate these attacks, we consider the χper function as three layers. Add-key, Non-linear (it is described with three AND and three XOR operation), and Mix-shift layers (see Figure 6). Note that to find a differential and linear characteristic the Add-key layer has no effect. Therefore, in these analyzes we can ignore it. Also, the action of the Non-linear layer can be described as parallel with a 3×3 S-box. This S-box in hexadecimal notation is given by Table 6.

x	0	1	2	3	4	5	6	7
$S(x)$	0	3	6	1	5	4	2	7

Table 6: The 3-bit S-box used in χper in hexadecimal form.

A.1 Differential/Linear Cryptanalysis

In order to argue for the resistance of χper against differential and linear attacks, we applied Mixed Integer Linear Programming (MILP) method as explained in [35,28,31] to search for differential and linear characteristics. The results are listed in Table 7 .

	♯ rounds	1	2	3	4	5	6	7	8
$w = 32$	Linear	1	3	6	11	19	24	28	32
	Differential	1	3	6	11	18	24	(32)	(37)

Table 7: Lowerbounds on the number of active S-boxes in χper. In case the MILP optimization was too long, we provide upper bounds between parentheses.

A.2 Impossible Differential characteristics

Impossible differential attack [23,5] finds two internal state differences Δ_i, Δ_o such that Δ_i is never propagated to Δ_o. The attacker then finds many pairs of plaintext/ciphertext and key values leading to (Δ_i, Δ_o). Those key values are wrong values, thus key space can be reduced. To search for impossible characteristics we applied MILP method based on the [11,30].

Our MILP model shows the longest impossible differential characteristics reach 6 rounds. The details of one of these characteristics can be seen in Table 8. Note that in this 8, the Input differential, Middle differential, and Output differential shows the differentials before S-box layer, after S-box layer, and after Mix-Shift layers, respectively. Also, the bits "0", "1", and "?" shows zero, active, and unknown differentials, respectively. To prove the impossibility of this differential, we use the following property that can be derived from the Differential Distribution Table (DDT) of χper S-box.

Fact 1 The S-box of χper has the following property:

- *If the input difference of the S-box is $0x1 = 001$, $0x2 = 010$, and $0x3 = 100$, then the output difference must be as $??1, ?1?$, and $1??$, respectively, where the $?$ shows an unknown difference bit.*

Round	Input differential	Middle differential	Output differential
	00000000000000000000000000000000	000000000000000?0000000000000000	000000000000000?0000010000000000
↓ 1	00000000000000000000000000000000	000000000000000?0000000000000000	00?0000000000?000001000000000000
	00000000000000010000000000000000	00000000000000010000000000000000	00000000?00000100000000000001?00
	000000000000000?0000010000000000	00?00000?0000???000?01000000??00	00??0000?0000???000?1?000?0???00
2	00?0000000000?000001000000000000	00?00000?0000???00010?000000??00	???00010?00???0???1?0?0?0???0000
	00000000?00000100000000000001?00	00?00000?0000?1?000?0?0000001?00	01??00???1???1???0?0???000????00
	00??0000?0000???000?1?000?0???00	????00??????????????????0?????00	
3	???00010?00???0???1?0?0?0???0000	????00??????????????????0?????00	
	01??00???1???1???0?0???000????00	????00???1??????????????0?????00	
		???????????????0????????????????	???0?????0?????0?????00????0?0??
		????????0???????????????????????	???0?????0?????0?????00????0?0??
		?????????0??????????????????????	???0?????0?????0?????00????0?0??
	???0?????0?????0?????00????0?0??	?0000????0?1???0?????00????000??	?0000??0000010000?0??00?0??0000?
3	???0?????0?????0?????00????0?0??	???0????00????00????00010?00?0??	?0000??00000?0000?0??0010??0000?
	???0?????0?????0?????00????0?0??	0??0?1???0?0???000???000???0000?	?0000??00000?0000?0??00?0??0000?
	?0000??0000010000?0??00?0??0000?	?0000??0000010000000000000?00001	000000?0000010000000000000000000
2	?0000??00000?0000?0??0010??0000?	00000000000000000?0?00010?000000	000000?00000?0000000000000000000
	?0000??00000?0000?0??00?0??0000?	00000000000000000000?00001?0000?	000000100000?0000000000000000000
	000000?0000010000000000000000000	00000000000000010000000000000000	00000000000000000000000000000000
↑ 1	000000?00000?0000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
	000000100000?0000000000000000000	00000010000000000000000000000000	00000000000000000001000000000000

Table 8: An impossible differential characteristic for 6 rounds χper when $w = 32$.

A.3 Zero-Correlation Linear Approximation

The zero-correlation attack is one of the cryptanalytic method introduced by Bogdanov and Rijmen [7]. The attack is based on linear approximations with zero correlation. To search for zero-correlation linear approximations, we applied the MILP method for χper. The longest zero-correlation linear approximation was obtained for 6 rounds of χper when $w = 32$. Table 9 shows an examples of this zero-correlation linear approximation. Note that in this table, the Input mask, Middle mask, and Output mask shows the linear masks before S-box layer, after S-box layer, and after Mix-Shift layers, respectively. Also, the bits "0", "1", and "?" shows zero, active, and unknown masks, respectively.

In the same way with impossible differential characteristics, Fact 1 is also true in linear mode and we have used it in Table 9.

Round	Input mask	Middle mask	Output mask
	00000000000000000000000000000000	0000?000000000000000000000000000	0000?0000000000000000?0000000000
↓ 1	00000000000000000000000000000000	0000?000000000000000000000000000	00000000000010000000000??0000000
	00001000000000000000000000000000	00001000000000000000000000000000	00000000000000000100000000000?00
	0000?0000000000000000?0000000000	0000?0000000?0000?000?0??0000?00	00?0?0?0??00?0?00?000?0??0000?00
2	00000000000010000000000??0000000	0000?00000000100000?000?0??0000?00	?000??00????000??00?00??1000?0?
	00000000000000000100000000000?00	0000?0000000?00001000?0??0000?00	00?0??0000?000?0??0000?00?000?10
	00?0?0?0??00?0?00?000?0??0000?00	0?0?0???0?????0?0??00??????000???	???0?????????0????0?????????????
3	?000??00????000??00?00??1000?0?	?0?0???0?????0?0??00??????000???	??????1????????0????????????????
	00?0??0000?000?0??0000?00?000?10	?0?0???0?????0?0??00??????000?1?	0???????0??1????????0?????0?0???
	??????0?????????????????????????	?0????0?0????0????00????1?0?0??0	?0?0??000?0??0?0??0000??10000100
3	??????0?????????????????????????	?00???00?0?0??0?0????0?0??0?0??1	?0?0??000?0?10?0??0000???0000?00
	??????0?????????????????????????	??????0?0????0????0?0????0??????	?0?0??00010??0?0??0000?1?0000?00
	?0?0??000?0??0?0??0000??10000100	00?0?000000000000?0000?010000100	0000?000000000000000000010000000
2	?0?0??000?0?10?0??0000???0000?00	?0000?00000010000?00000000000000	0000?0000000000000000000?0000000
	?0?0??00010??0?0??0000?1?0000?00	00?01000010?00?0?000000100000?00	00001000000000000000000?0000000
	0000?000000000000000000010000000	00000000000000000000000010000000	00000000000000000000000010000000
↑ 1	0000?0000000000000000000?0000000	00000000000000000000000000000000	00000000000000000000000000000000
	00001000000000000000000?0000000	00001000000000000000000000000000	00000000000000000000000000000000

Table 9: A zero-correlation linear approximation for 6 rounds χper when $w = 32$.

B Security Protocol Description Language model of the χperbp scheme

```
usertype Timestamp;
const XOR: Function;
const Concatenate: Function;
const Right: Function;
const Left: Function;
const Xper: Function;
hashfunction H;

  protocol Xperbp(Tag,Reader,CloudServer){
role Tag {
const IDi,Ki ;
var Mr,Mc;
fresh Rr : Nonce;
var Tr,Ts,Tt: Timestamp;
recv-!Tt(Tag,Tag,Tt);
recv-1(Reader,Tag,Rr);
macro Rt={Concatenate(Tt,Rr)}Ki;
macro RRt= Right(Rt);
macro Mt={XOR(IDi,Concatenate(Rr,RRt))}Ki;
send-2(Tag,Reader,Mt,RRt);
recv-5(Reader,Tag,Mc,Ts,Mr);
macro Mc'={IDi}XOR(Ki,Concatenate(Ts,RRt));
macro Mr'={XOR(Mt,Mc)}IDi;
match(Mc,Mc');
match(Mr,Mr');
claim(Tag,Secret,IDi);
claim(Tag,Secret,Ki);
claim(Tag,Niagree);
claim(Tag,Nisynch);
claim(Tag,Alive);
claim(Tag,Weakagree);
}
  role Reader {
const RID,Kr,Ki,IDi;
var RRt,RtR,RtL,Mc,Mt,MACc,DIi;
fresh Rr : Nonce;
var Tt,Ts,Tr: Timestamp;
recv-!Tr(Reader,Reader,Tr);
send-1(Reader,Tag,Rr);
recv-2(Tag,Reader,Mt,RRt);
macro MACr=H(Concatenate(Mt,RRt,Rr,Kr,Tr,RID));
send-3(Reader,CloudServer,MACr,Tr,Mt,Rr,RRt);
recv-4(CloudServer,Reader,Mc,DIi,Ts,MACc);
```

```
macro IDi'=XOR(DIi,RID,{Concatenate(Ts,Tr)}Kr);
macro MACc'=H(Concatenate(Mc,Mt,RRt,Rr,Ts,IDi',RID));
match(MACc,MACc');
macro Mr={XOR(Mt,Mc)}IDi;
send-5(Reader,Tag,Mc,Ts,Mr);
claim(Reader,Secret,IDi);
claim(Reader,Secret,Kr);
claim(Reader,Secret,RID);
claim(Reader,Niagree);
claim(Reader,Nisynch);
claim(Reader,Alive);
claim(Reader,Weakagree);
}
    role CloudServer{
const RID,Kr,IDi,Ki ;
var RRt,RtR,RtL,Mt,MACr,DIi;
fresh Rr : Nonce;
var Ts,Tt,Tr: Timestamp;
recv-!Ts(CloudServer,CloudServer,Ts);
recv-3(Reader,CloudServer,MACr,Tr,Mt,Rr,RRt);
macro Mt'={XOR(IDi,Concatenate(Rr,RRt))}Ki;
macro MACr'=H(Concatenate(Mt,RRt,Rr,Kr,Tr,RID));
match(Mt,Mt');
match(MACr,MACr');
macro Mc={IDi}XOR(Ki,Concatenate(Ts,RRt));
macro MACc=H(Concatenate(Mc,Mt,RRt,Rr,Ts,IDi,RID));
macro DIi=XOR(IDi,RID,{Concatenate(Ts,Tr)}Kr);
send-4(CloudServer,Reader,Mc,DIi,Ts,MACc);
claim(CloudServer,Secret,IDi);
claim(CloudServer,Secret,Ki);
claim(CloudServer,Secret,Kr);
claim(CloudServer,Secret,RID);
claim(CloudServer,Niagree);
claim(CloudServer,Nisynch);
claim(CloudServer,Alive);
claim(CloudServer,Weakagree);
} }
```