International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 5, October 2023, pp. 5462~5471
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i5.pp5462-5471

Secured authentication of radio-frequency identification system using PRESENT block cipher

Bharathi Ramachandra, Smitha Elsa Peter
Department of Electronics and Communication Engineering, Ponnaiyah Ramajayam Institute of Science and Technology, Deemed to be University, Tamilnadu, India

Article Info

Article history:

Received Jan 9, 2023
Revised Mar 21, 2023
Accepted Apr 7, 2023

Keywords:

Authentication
Field-programmable gate array
radio frequency identification system
Internet of things devices
PRESENT
Security

ABSTRACT

The internet of things (IoT) is an emerging and robust technology to interconnect billions of objects or devices via the internet to communicate smartly. The radio frequency identification (RFID) system plays a significant role in IoT systems, providing most features like mutual establishment, key establishment, and data confidentiality. This manuscript designed secure authentication of IoT-based RFID systems using the light-weight PRESENT algorithm on the hardware platform. The PRESENT-256 block cipher is considered in this work, and it supports 64-bit data with a 256-key length. The PRESENT-80/128 cipher is also designed along with PRESENT-256 at electronic codebook (ECB) mode for Secured mutual authentication between RFID tag and reader for IoT applications. The secured authentication is established in two stages: Tag recognition from reader, mutual authentication between tag and reader using PRESENT-80/128/256 cipher modules. The complete secured authentication of IoT-based RFID system simulation results is verified using the chip-scope tool with field-programmable gate array (FPGA) results. The comparative results for PRESENT block cipher with existing PRESENT ciphers and other light-weight algorithms are analyzed with resource improvements. The proposed secured authentication work is compared with similar RFID-mutual authentication (MA) approaches with better chip area and frequency improvements.

Corresponding Author:

Bharathi Ramachandra
Department of Electronics and Communication Engineering, Ponnaiyah Ramajayam Institute of Science and Technology, Deemed to be University
Tamilnadu, India
Email: bharavi_kumar@bmsit.in

1. INTRODUCTION

The internet of things (IoT) system is the most powerful technology, which can interconnect billions of devices via the internet for smart communication. The IoT system or infrastructure needs many devices or technologies (mobile communications, sensor networks, and embedded systems) to establish secure communication. The radio frequency identification (RFID) system is important in most IoT applications. The RFID system supports most IoT applications like healthcare, retail, home appliances, industries, transport, and many more. The RFID system supports data confidentiality, mutual authentications, and key establishment in most IoT applications [1], [2]. Any RFID system can perform object tracking, authentication, monitoring, and alert generation. The RFID system is caused by privacy and security issues. The security attacks concerned by the RFID system are forward and backward security, cloning, replay, relay, timing, eavesdropping, counter-felting, and tracking [3]. Many authentication protocols are available to secure the RFID system and solve privacy issues [4]. Authentication protocols like simple, fully-fledged, lightweight, and ultra-lightweight algorithms provide secured solutions to the RFID system [5]–[9]. The authentication protocol's main goal is to avoid the attacks mentioned above and provide better security services to the users.

Physical layer security (PLS) is an excellent way to keep emerging IoT applications safe. Many current methods for PLS to safeguard data and connection against attacks in IoT applications are available from previous research. Most PLS is software-based and cannot give cost-effective and reliable solutions. A few hardware-based solutions for PLS, such as microcontrollers and Raspberry Pi, suffer from hardware limits like latency and throughput. The lightweight-based cipher has been employed in previous hardware platform research for PLS, which uses more computational rounds for data and key updation, causing more confusion and consuming more resources on a single chip, degrading the overall system performance. As a result, a practical, lightweight algorithm is needed to protect IoT applications at the physical layer. In this Manuscript, the secured mutual authentication of an IoT-based RFID system (tag and reader) is designed using a light-weight PRESENT cipher on the FPGA platform. The contribution of the work is highlighted as follows:

The strong 256-bit Key is considered in the lightweight PRESENT cipher, which uses only 32 clock cycles for encryption/decryption operation and is the same as PRESENT-80/128 ciphers, with changes in the key updation unit (KUU). The electronic code block (ECB) mode functionality is considered in PRESENT-80/128/256 ciphers. The ECB mode architectures are easy to design and less complex in lower-constrained devices like IoT-based RFID systems. Secure authentication is established between IoT-based RFID Tag and Reader modules using PRESENT-80/128/256 at ECB mode with better hardware constraints.

The lightweight PRESENT-80/128/256 block cipher algorithm is discussed with architecture in section 2. Section 3 provides the secure authentication between the RFID tag and reader algorithm using PRESENT cipher at ECB mode. The results and discussion with comparison are covered in section 4. The overall work is concluded with futuristic scope in section 5.

This section explains the overview of the existing RFID-based mutual authentication using different application security algorithms. Songhela and Das [9] describe the RFID-MA protocol for its powerful privacy-preserving features. RFID tag identification is a key challenge, and preserving tag privacy is an issue. Public key cryptography algorithm like elliptic curve cryptography (ECC) is used for RFID mutual authentication. The MA work provides wide-weak, narrow-strong privacy under challenging situations, anti-cloning, and scalability features. The RFID mutual authentication also checks and analyzes the protocol formation, un-traceability for forward and backward, replay prevention, communication, and computational costs. Xiaohong and Yingmeng [10] explain the RFID-MA using the hash function. The hash function uses synchronously updated keys in the RFID-MA system. The self-synchronized methods and dynamic-tag updated key features are used in the MA protocol to perform the next verification process level. The design is modeled based on communication, security, and attack models. The performance of the security model is analyzed with replay attack, forward security, de-synchronization attack, and denial of service attack.

Mujahid *et al.* [11] describe the RFID-MA protocol using a new ultra-light-weight algorithm. The ultra-lightweight algorithm uses strong authentication and robust integrity features using recursive hash. The RFID-MA protocol uses basic logical and hash functions to support low-cost RFID tags. The protocol analyzes the integrity, confidentiality, and authentication features. The hardware architecture is also designed for RFID-MA using recursive hash [12] and highlights the FPGA and ASIC platform's resource utilization. Su *et al.* [13] explain the RFID-MA with universally composable (UC) features. The UC feature provides the most robust RFID-MA protocol security using public key encryption. The designs also analyze the RFID-active and passive tags characteristics with implementation. The work also presents the two public key updation protocols like trusted certifications and message authentication codes in public key infrastructure.

Kang *et al.* [14] present the RFID-MA protocol with a security analysis of the software environment. The work presents RFID-MA with a relay attack operation between tag and reader. The design protocol elaborates on the vulnerability attack effects and link timing analysis for the man-in-middle attack. The link timing provides the data rate values between tag and reader. Fan *et al.* [15] describe the RFID-based MA protocol using light-weight and ultra-weight algorithms in 5G applications for IoT devices. The cache concept is introduced in RFID-reader to reduce transmission and computational costs. If RFID-tags want to authenticate with reader, there is not much difference in computational cost. The protocol evaluates the security analysis with different attacks and analyzes performance concerning cost and computation time. Fan *et al.* [16] explain the cloud-based light-weight algorithm for RFID-MA protocol. Authenticating security in the cloud is a challenging task. The protocol uses simple logic encryption-based operation with a time-stamp to avoid the DoS and anti-synchronization attacks. The work provides detailed Burrows–Abadi–Needham (BAN) logical protocol proof. Luo and Liu [17] present an improved version of the RFID-MA protocol using the lightweight algorithm. The BAN logic formal analysis is improved to evaluate the security attacks and reduce computational resources (cost and storage). Xu *et al.* [18] and Zhu *et al.* [19] present the light-weight algorithm-based RFID MA protocol using physical unclonable function (PUF). The design includes tag identification, mutual authentication, and the updation process. The light-weight algorithm provides a detailed verification process for the single RFID-tag. The tag identifies the reader identified the Tag information in the tag identification process. The RFID-tag and reader are mutually authenticated in the MA process. The new key is updated to verify the tag and reader process in the updation stage. The work analyzes the computational and storage cost of RFID tags and reader. Ibrahim and Dalkılıç [20] present advanced encryption standards (AES) and ECC algorithms based on MA protocol designed for RFID systems and proved on wireless identification and sensing platform (WISP). The work includes the AES and ECC algorithms incorporated into the RFID tag-reader process. The results are discussed for communication cost and response time with comparison. Naeem *et al.* [21] explain the ECC-based scalable RFID-MA protocol for the IoT environment. The work suggests and corrects the scalability problems. The protocol describes the initialization and authentication process in detail using ECC. The results also check with the BAN logic process for authentication proof.

2. LIGHT-WEIGHT PRESENT-80/128/256 ALGORITHMS

In this work, the lightweight PRESENT block cipher is designed with different key sizes like 80-bit, 128-bit, and 256-bit. The PRESENT cipher is simple, easy to design, and uses arithmetic-logical and shifting operations to construct the hardware architecture. The PRESENT-algorithm mainly performs three operations: i) encryption, ii) decryption, and iii) key updation unit (KUU) operations. The PRESENT encryption/decryption operation processes parallel with KUU, reducing computational complexity and chip area utilization. The list of abbreviations used in this work is tabulated in Table 1.

Table 1. List of abbreviations used in this work

Parameter	Description	Parameter	Description
ENC	Encryption	K_S	Temporary key register
DEC	Decryption	UKR_out	Updated key register output
IT	Input text	IUKR_out	Inverse updated key register output
CT	Cipher text	P_Layer	Permutation layer
OT	Output text	PL_out	Permutation layer output
K	Key input	R_C	Round counter
KUU	Key updation unit	Sbox_in	SBOX-input
IKUU	Inverse Key updation unit	Sbox_out	SBOX-output
SBOX	Substitution-Box	R_A	Reader authentication
RNG64	64-bit random number	T_A	Tag authentication
ld	Load signal	R_1, R_2, and R_3	Random numbers
kld	Key load signal	Ch_1, Ch_2	Reader challenge -1 and 2
rst	Reset signal	Tag_Ch_1, Tag_Ch_2	Tag challenge data 1 and 2
U_K	Updated key	RS_1, RS_2	Reader side data -1 and 2
Key_Reg	Key register	RR_1, RR_2	Reader Response -1 and 2

2.1. PRESENT-80/128/256 encryption operation

The PRESENT algorithm is a block cipher light-weight cryptographic algorithm that works with 64-bit data size and variable keys (80/128). In this work, the 256-bit key operation is considered additionally for both PRESENT encryption/decryption operations to strengthen the security in IoT devices. The PRESENT-80/128/256 algorithm uses 32 rounds for either encryption or decryption. The PRESENT architecture is designed in a reconfigurable manner to support 16 or 64 rounds of operations for both encryption/decryption operations. The PRESENT-80/128/256 design process is represented in algorithm 1.

The algorithm mainly has three stages of operations: initialization stage, round operation, key process operation in parallel, and output stage. The PRESENT encryption process has 64-bit input text (IT), 80/128/256-bit key (K), and control signals like load (ld) and key load (kld) at the input side. The 64-bit cipher text (CT), 80/128/256-bit updated_key (U_K), is obtained at the output side.

First, initialize the registers (64-bit State, 80/128/256-bit Key_Reg) by performing the reset (rst) operation. If the load (ld) is activated, then consider input text (IT); else, permutation layer output (PL_out) as state output. Similarly, If the key load (kld) is activated, then consider key (K); else, updated key register output (UKR_out) as key register (Key_Reg) output. In the next stage, perform round operations that contain XOR, S-BOX, and permutation layer operations. The 5-bit round counter (R_C) counts the number of rounds. First, complete the XOR operation of the state register with Key_Reg and store the results in Sbox_in. The SBOX operation followed by the permutation layer (PL_out) operation is performed to generate each round operation output. After the 31st round, The R_C is reset to zero. Similarly, the key process is also operating parallel with the round operation. The key updation unit updates the key register (Key_Reg) data for every round and stores the results in the updated key register output (UKR_out). The operation continues till the 31st round and the next stops. In the output stage, consider the key register output as the final 80/128/256-bit updated_key (U_K) and state register data as the final 64-bit cipher text (CT). Consider a 4-bit/ 6-bit round counter (R_C) to perform 16/64 rounds operation in PRESENT encryption using 80/128/256-bit Key. Bogdanov *et al.* [22] consider the SBOX operation and the Permutation layer activities. The PRESENT decryption process inputs the Ciphertext and the 80/128/256-bit updated_key. The KUU process accepts 80/128/256-bit key register and 5-bit round counter values as input.

Algorithm 1. PRESENT-80/128/256 encryption
```
Input: 64-bit Input Text (IT), 80/128/256-bit Key (K), Load (ld), Key Load (kld);
Output: 64-bit Cipher Text (CT), 80/128/256-bit Updated _Key (U_K);
  1. Initialization:
     a. Define Registers: 64-bit State, 80/128/1256-bit Key_Reg;
     b. If (ld =1) then State=IT else State=PL_out;
     c. If (kld =1) then Key_Reg=Key (K) else Key_Reg=UKR_out;
  2. Round Operation:
                For (i=0; i<31; i= i +1;
     a. XOR operation: Sbox_in=State XOR Key_Reg;
     b. S-BOX operation: Sbox_out=SBOX(Sbox_in);
     c. Permutation layer operation: PL_out=P_Layer (Sbox_out);
     d. Repeat the (a-c) operation till 31 rounds using Round Counter (R_C).
                End Loop
  3. Key process:
                For (i=0; i<31; i= i +1;
     a. Perform Key updation operation:
                Updated Key Register output (UKR_out)=KUU (Key_Reg);
                End Loop
  4. Output Stage:
     a. After the 31st Round: Encryption outputs are generated
            -  CT=State;
            -  U_K=Key_Reg;
```

2.2. PRESENT-80/128/256 decryption operation

The PRESENT-80/128/256 decryption process is the same as the encryption process with few changes. The inverse SBOX and inverse Permutation layer operations are used around the operation and inverse key updation unit (IKUU) operation. After the 31st round, the State register generates the 64-bit output text (OT), and the inverse key register generates the 80/128/256-bit inverse updated_key register (IUKR_out) output.

2.3. PRESENT-80/128/256 key updation operation

The PRESENT-256 key updation operation is represented in algorithm 2. The key updation unit (KUU) and inverse KUU (IKUU) are used for encryption and decryption. For the PRESENT-256 encryption algorithm, initially, the key register (Key_Reg) performs a left shift operation 195 times, stores the results in the temporary register (K_S), and then processes the key updation operation. In the updation process, perform XOR operation for 5-bit round counter (R_C) within temporary register K_S [129:125]. Perform two 4-bit SBOX operations for temporary register K_S [251:248] and K_S [255:252] inputs. Finally, concatenate all the temporary register K_S values to form the 256-bit Updated Key Register output (UKR out). For the PRESENT-256 Decryption algorithm, perform an inverse updation process followed by shift operations. The inverse updation process is the same as the encryption updation process, changing the SBOX operations with Inverse SBOX operations and storing the results in temporary register K_S. Finally, perform left shift operation (<< 61) times with K_S to generate inverse updated _key register (IUKR_out) output.

The Key updation operation for 80-bit and 128-bit is similar to 256-bit KUU and IKUU operations with few changes. Instead of 256-bit, use the corresponding 80-bit or 128-bit for KUU operations. For the PRESENT-80 and 128 encryption process, change left shift operations to 19 and 67 times. To perform the XOR operation, use a 5-bit and 6-bit round counter for PRESENT-80 and 128 in the encryption/decryption process. Perform only one-time SBOX and two S-BOX functions for PRESENT-80 and PRESENT 128, respectively, in the encryption/decryption process. For IKUU operation, use the same left shift (<<61) operation in PRESENT -80 and 128 modules.

Algorithm 2. PRESENT-256 key updation unit (KUU) and inverse KUU operation
```
Input: 256-bit Key register (Key_Reg), 7-bit Round counter (R_C);
Output: 256-bit Updated Key Register output (UKR_out) and inverse Updated _Key Register
(IUKR_out) output.
    1.Encryption Process:
```

```
    a. Perform Shift left Operation: K_S=Key_Reg << 195;
    b. Updation Process:
       -  UKR_out [123:0]=K_S [123:0];
       -  UKR_out [128:124]=K_S [128:124] XOR R_C;
       -  UKR_out [247:129]=K_S [247:129];
       -  UKR_out [251:248]=SBOX (K_S [251:248]);
       -  UKR_out [255:252]=SBOX (K_S [255:252]);
 2.Decryption Process:
    c. Inverse Updation Process:
       -  K_S [123:0]=Key_Reg [123:0];
       -  K_S [128:124]=Key_Reg [128:124] XOR R_C;
       -  K_S [247:129]=Key_Reg [247:129];
       -  K_S [251:248]=Inverse_SBOX (Key_Reg [251:248]);
       -  K_S [255:255]=Inverse_ SBOX (Key_Reg [255:252]);
    d. Perform Shift left Operation: IUKR_out=K_S << 61;
```

2.4. PRESENT-80/128/256 at ECB mode

The electronic codebook (ECB) mode is one of the block cipher modes to provide confidentiality by performing encryption or decryption of one fixed-length block. The ECB mode offers the simplest form of encryption by dividing the input text into blocks, and each block performs encryption or decryption separately. The PRESENT-80/128/256 algorithm at ECB mode performs mutual authentication between Tag and Reader for IoT-based RFID systems. At EBC mode, The PRESENT -80/128/256 algorithm consists of two 64-bit Input Text (IT_1, IT_2), 80/128/256-bit Key (K) at the input side, and two 64-bit cipher texts (CT_1, CT_2), 64-bit Output Text (OT_1, OT_2) at the output side. The PRESENT 80/128/256 algorithm performs the encryption (ENC) and decryption (DEC) at ECB mode as in (1) and (2), respectively. The two times of PRESENT-80/128/256 encryption or decryption are performed parallel to generate corresponding ciphertexts or output texts.

$$Encryption\, process: Cipher\ text\ (CT1, CT2) = ENC\ (IT1, IT2)\ using\ Key\ (K); \quad (1)$$

$$Decryption\, process: Output\ text\ (OT1, OT2) = DEC\ (CT1, CT2)\ using\ same\ Key\ (K); \quad (2)$$

3. SECURED AUTHENTICATION OF IoT-BASED RFID SYSTEM

The RFID system is the default option in most cases to communicate between two or more IoT devices. The secured authentication is established mutually between RFID tag and reader using light-weight PRESENT-80/128/256 cipher for IoT-based system. The secured authentication is specified in two stages: i) tag recognition from reader and ii) secured mutual authentication between tag and reader using a PRESENT cipher.

In the 1st stage, the reader requests the tag for communication establishment. The tag module will not accept the reader easily and generates the identification data using a random number. The tag module performs PRESENT encryption for identification data and sends the encrypted form data to the reader module. The reader module performs a PRESENT decryption operation and generates the reader's data. If the reader's data and identification data match, then the tag module is recognized from the reader module.

The secured authentication (SA) between tag and reader is performed for the IoT-based RFID system, represented in Figure 1 and algorithm 3. The SA operation is processed only after verifying the tag from reader. The SA of an IoT-based RFID system mainly contains an 80/128/256-bit key at the input side and 1-bit reader authentication (R_A), tag authentication (T_A) signals outside. Define the 64-bit three random numbers (R_1, R_2, and R_3) to process tag and reader SA operation. The reader sends a communication request for the tag module. The tag responds with a Random number (R_1). The reader receives the R_1 and uses one more random number (R_2) to create a challenge operation ($Ch_{1,2}$) by performing the PRESENT-80/128/256 decryption operation.

The tag module performs the PRESENT-80/128/256 encryption operation using the challenge data ($Ch_{1,2}$) and generates the tag challenge data ($Tag\_Ch_{1,2}$). If the Tag_Ch_2 equals the R_1 data, the reader is authenticated successfully. The tag is ready to respond to the reader by performing PRESENT-80/128/256 encryption operation with Tag_Ch_1 and R_1 and storing the results as $RS_{1,2}$. The reader starts performing the PRESENT-80/128/256 decryption operation using $RS_{1,2}$ and stores the results as $RR_{1,2}$. If the RR_1 equals the R_2 data, the tag is authenticated successfully. Once both the tag and reader are authenticated, then mutual communication is established for further communication.

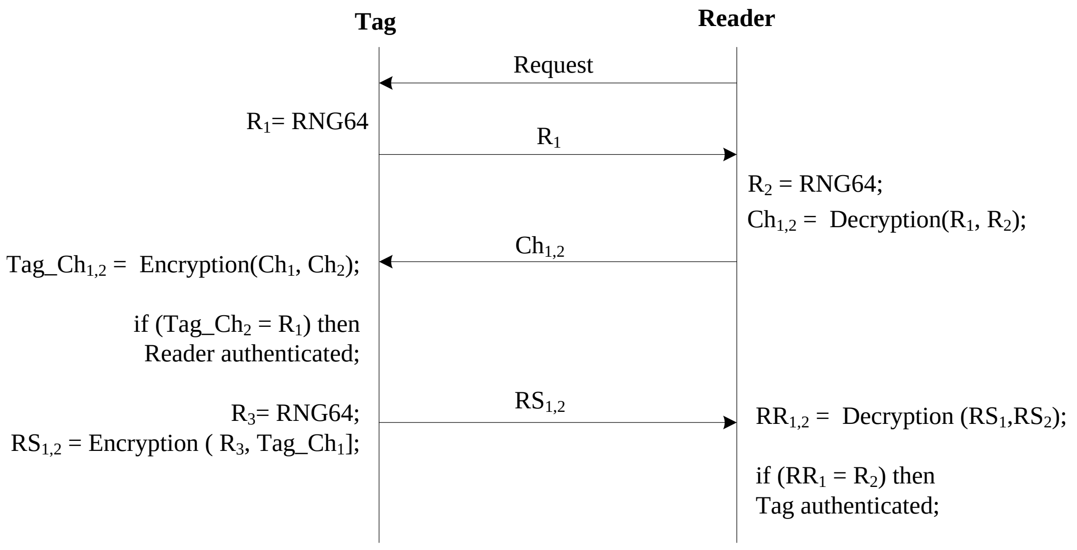
Tag
Reader
Request
R_1= RNG64
R_1
R_2 = RNG64;
$Ch_{1,2}$ = Decryption(R_1, R_2);
$Ch_{1,2}$
$Tag\_Ch_{1,2}$ = Encryption(Ch_1, Ch_2);
if ($Tag\_Ch_2 = R_1$) then
Reader authenticated;
R_3= RNG64;
$RS_{1,2}$ = Encryption (R_3, Tag_Ch_1];
$RS_{1,2}$
$RR_{1,2}$ = Decryption (RS_1,RS_2);
if ($RR_1 = R_2$) then
Tag authenticated;

Figure 1. Secured authentication of IoT-based RFID system between tag and reader

Algorithm 3. Secured authentication of IoT-based RFID system using PRESENT algorithm at ECB mode

```
Input: 80/128/256- bit Key;
Output: 1-bit reader authentication (R_A), tag authentication (T_A)
   A. Tag Side:
           a. Request from Reader: Generate Random Number $R_1$=RNG64;
   B. Reader Side:
           a. Generate Random Number $R_2$= RNG64;
           b. Perform Decryption Operation: $Ch_{1,2}$=PRESENT_Decryption ($R_1$, $R_2$);
   C. Tag Side:
           a. Perform Encryption Operation: $Tag\_Ch_{1,2}$=PRESENT_Encryption ($Ch_1$, $Ch_2$);
           b. If ($Tag\_Ch_2$=$R_1$) Then Reader Authenticated
           c. Generate Random Number $R_3$=RNG64;
           d. Perform Encryption Operation: $RS_{1,2}$=PRESENT_Encryption ($R_3$, $Tag\_Ch_1$);
   D. Reader Side:
           a. Perform Decryption Operation: $RR_{1,2}$=PRESENT_Decryption ($RS_1$, $RS_2$);
           If ($RR_1$=$R_2$) Then Tag Authenticated
```

4. RESULTS AND DISCUSSION

The secure authentication (SA) of IoT-based RFID system results is analyzed in this section. The discussion includes simulation and FPGA verified results, synthesis results, and comparative discussion with existing similar approaches. The secure authentication of the IoT-based RFID system is simulated on ModelSim 6.5c simulator and verified on results on Artix-7 using the Chipscope-pro tool. The PRESENT-80/128/256 modules, PRESENT cipher at ECB mode, and SA of IoT-based RFID system utilization summary are obtained after synthesis operation using Artix-7 FPGA. The simulation results of SA of IoT-based RFID system are represented in Figure 2. The PRESENT-128 at ECB mode is considered to block cipher for SA simulation between RFID tag and reader. The global clock (clk) is activated with an active-low reset (rst). Initially, define three 64-bit random numbers (RN1, RN2, and RN3) used in SA between tag and reader. The control signals like key load (kld1 and kld2) and load (ld1 and ld2) are active low to start the SA process.

After performing the PRESENT-128 decryption operation, the reader challenge values (Reader_cha1 and Reader_Cha2) are obtained. The tag accepts the challenges and performs the PRESENT-128 encryption operation to obtain the two tag challenge values (Tag_Cha1 and Tag_Cha2). The reader is authenticated if the random number (RN1) equals the Tag_Cha1 value (Reader_Auth). Perform PRESENT-128 encryption with tag values and obtains the two tag responses (Tag_Resp1 and Tag_Resp2) to establish tag authentication. The RFID reader responds to the tag response value by performing the PRESENT-128 decryption operation and generates the reader response values (Reader_Resp1 and Reader_Resp2). The tag is authenticated if the random number (RN2) equals the Reader_Resp1 value (Tag_Auth). So mutual authentication between RFID tag and reader is established for further communication securely. The SA of an IoT-based RFID system takes 1.285 µs to complete the authentication process between Tag and Reader.

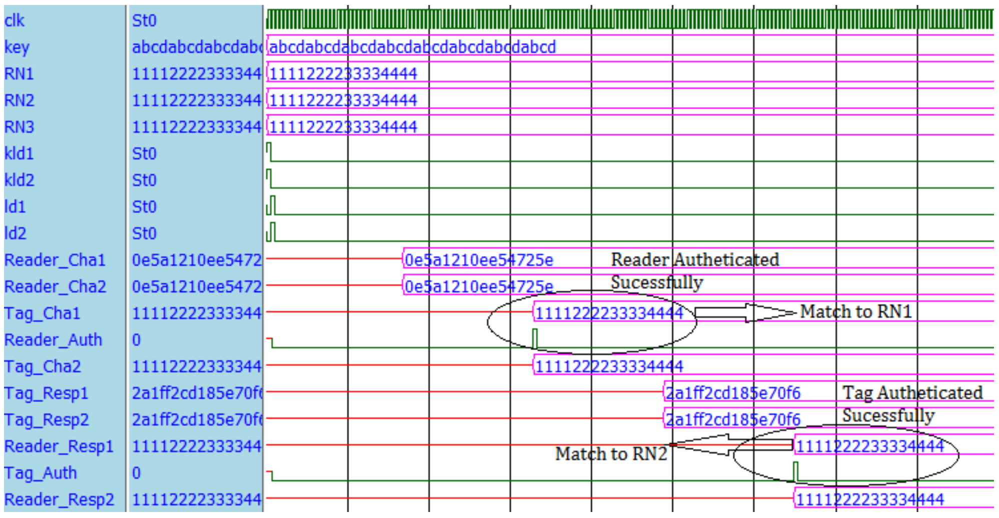
clk St0
key abcdabcdabcdabc abcdabcdabcdabcdabcdabcdabcdabcd
RN1 11112222333344 1111222233334444
RN2 11112222333344 1111222233334444
RN3 11112222333344 1111222233334444
kld1 St0
kld2 St0
ld1 St0
ld2 St0
Reader_Cha1 0e5a1210ee5472 0e5a1210ee54725e
Reader_Cha2 0e5a1210ee5472 0e5a1210ee54725e
Reader Autheticated Sucessfully
Tag_Cha1 11112222333344 1111222233334444
Match to RN1
Reader_Auth 0
Tag_Cha2 11112222333344 1111222233334444
Tag_Resp1 2a1ff2cd185e70f 2a1ff2cd185e70f6
Tag_Resp2 2a1ff2cd185e70f 2a1ff2cd185e70f6
Tag Autheticated Sucessfully
Reader_Resp1 11112222333344 1111222233334444
Match to RN2
Tag_Auth 0
Reader_Resp2 11112222333344 1111222233334444

Figure 2. Simulation results of secured authentication of IoT-based RFID system

The synthesis results provide the resource utilization report of the proposed work. The resources utilized for PRESENT-80/128/256 light-weight block ciphers on Artix-7 FPGA are tabulated in Table 2. The PRESENT cipher uses 64-bit data as input and performs 32 rounds of encryption and decryption with three different keys 80-bit, 128-bit, and 256-bit. The PRESENT-80 cipher consumes 359 slices and 437 LUTs and utilizes 0.187 W total power on Artix-7 FPGA. Similarly, the PRESENT-128 cipher consumes 455 slices, 489 LUTs, and uses 0.189W total power.

Table 2. Resources utilized on Artix-7 for PRESENT-80/128/256 ciphers

Hardware resources	PRESENT-80	PRESENT-128	PRESENT-256
Data Size/Key/Rounds	64/80/32	64/128/32	64/256/32
Slice registers	359	455	711
Slice LUTs	437	489	617
Max. frequency (MHz)	410.7	410.7	410.7
Total power (W)	0.187	0.189	0.191
Throughput (Mbps)	822	822	822
Efficiency (Mbps/slice)	2.29	1.8	1.15

The PRESENT-256 cipher consumes 711 slices and 617 LUTs and utilizes 0.191 W total power on Artix-7 FPGA. The PRESENT-80/128/256 ciphers work at 410.7 MHz frequency and obtain the Throughput of 822 Mbps on Artix-7 FPGA. The hardware efficiency is calculated based on obtained Throughput per slice. The PRESENT-80/128/256 cipher receives 2.29, 1.8, and 1.15 Mbps/slice hardware efficiency on Artix-7 FPGA. If the key size of the PRESENT cipher increases, the resource utilization (Area (Slices, LUTs) and power) is increased by decreasing the hardware efficiency.

The resource utilization results are obtained after place, and route operation for secure authentication of IoT-based RFID system using PRESENT-80/128/256 at ECB mode is tabulated in Table 3. The SA of the IoT-based RFID system utilizes 754 slices, 1086 LUTs using PRESENT-80, 945 slices, 1276 LUTs using PRESENT-128, 1457 slices, and 1783 LUTs using PRESENT-256 cipher at ECB mode. The SA of the IoT-based RFID system consumes a total power of 0.107 W, 0.112 W, and 0.114 W using PRESENT-80/128/256.

Table 3. Implementation results for secured authentication of IoT-based RFID system on Artix-7

Hardware resources	SA using PRESENT-80 (ECB Mode)	SA using PRESENT-128 (ECB Mode)	SA using PRESENT-256 (ECB Mode)
Slice registers	754	945	1457
Slice LUTs	1086	1276	1783
Max. frequency (MHz)	403.112	403.112	403.112
Total Power (W)	0.107	0.114	0.12

The performance parameters comparison of light-weight block cipher PRESENT-80/128 (only encryption) operation with existing similar PRESENT ciphers [23]–[26] is tabulated in Table 4. The data size is fixed to 64-bit with variable keys (80-bit and 128-bit) compared to FPGA (Virtex-5 and Artix-7) devices. The chip area (Slices, LUTs, and FFs), maximum frequency (Fmax), latency in terms of clock cycles (CC), Throughput, and efficiency parameters are considered for comparison. The existing PRESENT [23]–[25] ciphers are implemented on Virtex-5 FPGA, which consumes more latency (clock cycles) and obtains less Throughput (Mbps) than the proposed PRESENT-80/128 ciphers. The Throughput is estimated using data size, latency, and maximum frequency parameters. The existing PRESENT [26] is implemented on Artix-7, which consumes more chip area (Slices, LUTs, and FFs), less operating frequency, moderate throughput, and more efficiency than the proposed PRESENT-80 cipher.

Table 4. Performance comparison of PRESENT cipher with existing PRESENT approaches [23]–[26]

PRESENT Ciphers	Ref [23]	Ref [23]	Ref [24]	Ref [24]	Ref [25]	Ref [25]	Ref [26]	This Work	This Work
Data size	64	64	64	64	64	64	64	64	64
Key	80	128	80	128	80	128	80	80	128
FPGA device	Virtex-5	Virtex-5	Virtex-5	Virtex-5	Virtex-5	Virtex-5	Artix-7	Virtex-5	Virtex-5
FF's	153	201	137	137	149	197	213	183	231
LUTs	190	239	348	396	216	267	426	272	324
Slices	67	73	126	150	54	72	213	183	231
Fmax (MHz)	542.3	431.78	215.42	212.13	545.05	433.78	368	455.76	455.76
Latency (CC)	133	136	33	33	51	63	32	32	32
Throughput (Mbps)	260.96	203.19	417.79	411.41	683.98	440.66	736	911.53	911.53
Efficiency (Mbps/Slice)	3.89	2.78	3.31	2.74	12.66	6.12	3.45	4.98	3.94

The proposed PRESENT-80/128 cipher with existing lightweight cipher approaches [27]–[30] are compared concerning the performance resources on Spartan-3 FPGA, tabulated in Table 5. The existing lightweight ciphers like SPECK [27], SIMON [27], XTEA-1 [28], XTEA-3 [28], and LED [29], [30] are considered for comparison. The 64-bit data and 128-bit keys are used in all the existing ciphers [27]–[30], and all are implemented on Spartan -3 FPGA devices. The area (slices), throughput (Mbps), and hardware efficiency parameters are considered for comparison. The proposed PRESENT-80/128/256 ciphers obtain better throughput (Mbps) and efficiency (Kbps/slice) than the existing lightweight ciphers [27]–[30] on Spartan -3 FPGA.

Table 5. Performance Comparison of PRESENT-80/128 cipher with existing light-weight cipher approaches

Light-weight ciphers	Data size	Key size	FPGA Device	Area (slices)	Throughput (Mbps)	Efficiency (Kbps/Slice)
SPECK [27]	64	128	Spartan -3	24	9.6	400
SIMON [27]	64	128	Spartan -3	34	2.8	82.35
XTEA-1 [28]	64	128	Spartan -3	266	19	71.42
XTEA-3 [28]	64	128	Spartan -3	254	36	141.73
LED [29]	64	128	Spartan -3	315	105.34	334.41
LED [30]	64	128	Spartan -3	404	478	1183
PRESENT -80	64	128	Spartan -3	181	391.54	2163.23
PRESENT -128	64	128	Spartan -3	231	391.54	1694.97
PRESENT -256	64	256	Spartan -3	284	391.54	1378.66

The resource comparison of SA-based IoT-based RFID systems with an existing similar approach [31] is tabulated in Table 6. The current mutual authentication methods of RFID systems use lightweight ciphers like Hummingbird (HB), XTEA, and PRESENT to compare the same Spartan -3E FPGA device. The RFID-MA using HB [31] uses a 16-bit data size, and a 256-bit key utilizes 2,411 slice-FFs and works at 40.1 MHz. The proposed SA of an IoT-based RFID system operates fewer slice-FFs and works at a better frequency than the existing RFID-MA approach using Hummingbird (HB), XTEA, and PRESENT ciphers [31]. The proposed PRESENT-80/128/256 encryption/decryption operation works parallel with the key updation mechanism, which reduces the execution time (Latency). The encryption and decryption operations are designed with simple XOR, logical operations, and sequential circuits, giving better throughput and reducing the hardware complexity.

Table 6. Resource comparison of SA based IoT based RFID system with existing similar approach [31]

Secured authentication methods	Data size	Key size	FPGA Device	Slice-FFs	Frequency (MHz)
RFID-MA using HB [31]	16	256	Spartan-3E	2,411	40.1
RFID -MA using XTEA [31]	64	128	Spartan-3E	2,603	60.6
RFID -MA using PRESENT [31]	64	128	Spartan-3E	2,553	59.16
Proposed work	64	128	Spartan-3E	996	192.658

5. CONCLUSION AND FUTURE WORK

The lightweight PRESENT-80/128/256 block cipher algorithm is used to develop an efficient and cost-effective secure authentication mechanism for IoT-based RFID systems. In electronic codebook (ECB) mode, the PRESENT-80/128/256 block cipher performs block-wise encryption and decryption. The RFID Tag and Reader are authenticated securely for data communication using the PRESENT-80/128/256 block cipher. The simulation findings are functionally confirmed when compared to FPGA results. The synthesis results are tabulated for the PRESENT-80/128/256 block cipher and its secured authentication modules. On the Artix-7 FPGA, the secured authentication protocol uses 2% slices, runs at 403.1 MHz, and costs 0.12 W total power. The PRESENT-80/128/256 cipher outperforms similar PRESENT ciphers and other lightweight algorithms regarding chip area and throughput. The secure authentication of IoT-based RFID system using PRESENT is compared with existing RFID-MA protocols with a reduction of 50% in slice-FFs and 68% in operating frequency. The PRESENT-80/128/256 cipher is used further to analyze the security by concerning the different attacks to know the data/key availability and complexity.

REFERENCES

[1] S. Piramuthu, “RFID mutual authentication protocols,” *Decision Support Systems*, vol. 50, no. 2, pp. 387–393, Jan. 2011, doi: 10.1016/j.dss.2010.09.005.

[2] R. Baashirah and A. Abuzneid, “Survey on prominent RFID authentication protocols for passive tags,” *Sensors*, vol. 18, no. 10, Oct. 2018, doi: 10.3390/s18103584.

[3] S. M. Mohsin, I. A. Khan, S. M. Abrar Akber, S. Shamshirband, and A. T. Chronopoulos, “Exploring the RFID mutual authentication domain,” *International Journal of Computers and Applications*, vol. 43, no. 2, pp. 127–141, Feb. 2021, doi: 10.1080/1206212X.2018.1533614.

[4] A. Ibrahim and G. Dalkılıc, “Review of different classes of RFID authentication protocols,” *Wireless Networks*, vol. 25, no. 3, pp. 961–974, Apr. 2019, doi: 10.1007/s11276-017-1638-3.

[5] U. Mujahid, G. Unabia, H. Choi, and B. Tran, “A review of ultralightweight mutual authentication protocols,” *International Journal of Electrical and Computer Engineering*, vol. 14, no. 4, pp. 96–101, 2020.

[6] J.-S. Chou, Y. Chen, C.-L. Wu, and C.-F. Lin, “An efficient RFID mutual authentication scheme based on ECC,” *Cryptology ePrint Archive*, pp. 1–20, 2011.

[7] H. Kim, “Enhanced hash-based RFID mutual authentication protocol,” in *Communications in Computer and Information Science*, vol. 339, Springer Berlin Heidelberg, 2012, pp. 70–77.

[8] M. L. Das, “Strong security and privacy of RFID system for internet of things infrastructure,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8204, Springer Berlin Heidelberg, 2013, pp. 56–69.

[9] R. Songhela and M. L. Das, “Yet another strong privacy-preserving RFID mutual authentication protocol,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8804, Springer International Publishing, 2014, pp. 171–182.

[10] Z. Xiaohong and H. Yingmeng, “RFID mutual-authentication protocol with synchronous updated-keys based on hash function,” *The Journal of China Universities of Posts and Telecommunications*, vol. 22, no. 6, pp. 27–35, Dec. 2015, doi: 10.1016/S1005-8885(15)60690-2.

[11] U. Mujahid, M. Najam-ul-Islam, and M. A. Shami, “RCIA: a new ultralightweight RFID authentication protocol using recursive hash,” *International Journal of Distributed Sensor Networks*, vol. 11, no. 1, Jan. 2015, doi: 10.1155/2015/642180.

[12] U. Mujahid, A. R. Jafri, and M. Najam-Ul-Islam, “Efficient hardware implementation of ultralightweight RFID mutual authentication protocol,” *Journal of Circuits, Systems and Computers*, vol. 25, no. 7, 2016, doi: 10.1142/S021812661650078X.

[13] C. Su, B. Santoso, Y. Li, R. Deng, and X. Huang, “Universally composable RFID mutual authentication,” *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 1, pp. 1–1, 2015, doi: 10.1109/TDSC.2015.2434376.

[14] Y. S. Kang, E. O’Sullivan, D. Choi, and M. O’Neill, “Security analysis on RFID mutual authentication protocol,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9503, Springer International Publishing, 2016, pp. 65–74.

[15] K. Fan, Y. Gong, C. Liang, H. Li, and Y. Yang, “Lightweight and ultralightweight RFID mutual authentication protocol with cache in the reader for IoT in 5G,” *Security and Communication Networks*, vol. 9, no. 16, pp. 3095–3104, Nov. 2016, doi: 10.1002/sec.1314.

[16] K. Fan, Q. Luo, K. Zhang, and Y. Yang, “Cloud-based lightweight secure RFID mutual authentication protocol in IoT,” *Information Sciences*, vol. 527, pp. 329–340, Jul. 2020, doi: 10.1016/j.ins.2019.08.006.

[17] L. Luo and D. Liu, “An improved lightweight RFID mutual-authentication protocol,” *Proceedings of the Second International Conference on Mechanics, Materials and Structural Engineering* (*ICMMSE 2017*), 2017, doi: 10.2991/icmmse-17.2017.45.

[18] H. Xu, J. Ding, P. Li, F. Zhu, and R. Wang, “A lightweight RFID mutual authentication protocol based on physical unclonable function,” *Sensors*, vol. 18, no. 3, Mar. 2018, doi: 10.3390/s18030760.

[19] F. Zhu, P. Li, H. Xu, and R. Wang, “A lightweight RFID mutual authentication protocol with PUF,” *Sensors*, vol. 19, no. 13, Jul. 2019, doi: 10.3390/s19132957.

[20] A. Ibrahim and G. Dalkılıç, "An advanced encryption standard powered mutual authentication protocol based on elliptic curve cryptography for RFID, proven on WISP," *Journal of Sensors*, vol. 2017, pp. 1–10, 2017, doi: 10.1155/2017/2367312.
[21] M. Naeem, S. A. Chaudhry, K. Mahmood, M. Karuppiah, and S. Kumari, "A scalable and secure RFID mutual authentication protocol using ECC for internet of things," *International Journal of Communication Systems*, vol. 33, no. 13, Sep. 2020, doi: 10.1002/dac.3906.
[22] A. Bogdanov *et al.*, "PRESENT: an ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems*, vol. 4727, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466.
[23] C. A. Lara-Nino, A. Diaz-Perez, and M. Morales-Sandoval, "Lightweight hardware architectures for the present cipher in FPGA," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 9, pp. 2544–2555, Sep. 2017, doi: 10.1109/TCSI.2017.2686783.
[24] J. G. Pandey, T. Goel, and A. Karmakar, "A high-performance and area-efficient VLSI architecture for the PRESENT lightweight cipher," in *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*, Jan. 2018, pp. 392–397, doi: 10.1109/VLSID.2018.96.
[25] V. K. Reddy, R. Surya, A. Reddy, and P. S. Kumar, "FPGA implementation of present algorithm with improved security," in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, Jun. 2019, pp. 404–409, doi: 10.1109/ICECA.2019.8821908.
[26] A. Varici *et al.*, "Fast and efficient implementation of lightweight crypto algorithm PRESENT on FPGA through processor instruction set extension," in *2019 IEEE East-West Design and Test Symposium (EWDTS)*, Sep. 2019, pp. 1–5, doi: 10.1109/EWDTS.2019.8884397.
[27] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "SIMON and SPECK: block ciphers for the internet of things," *Cryptology ePrint Archive*, 2015.
[28] J.-P. Kaps, "Chai-Tea, cryptographic hardware implementations of xTEA," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5365, Springer Berlin Heidelberg, 2008, pp. 363–375.
[29] R. RajaRaja and D. Pavithra, "Implementation of hardware efficient light weight encryption method," in *2013 International Conference on Communication and Signal Processing*, Apr. 2013, pp. 191–195, doi: 10.1109/iccsp.2013.6577041.
[30] S. Guruprasad and B. Chandrasekar, "An evaluation framework for security algorithms performance realization on FPGA," in *2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, Feb. 2018, pp. 1–6, doi: 10.1109/ICCTAC.2018.8370396.
[31] S. Seshabhattar, S. K. Jagannatha, and D. W. Engels, "Security implementation within GEN2 protocol," in *2011 IEEE International Conference on RFID-Technologies and Applications*, Sep. 2011, pp. 402–407, doi: 10.1109/RFID-TA.2011.6068669.

BIOGRAPHIES OF AUTHORS

Bharathi Ramachandra has got her B.E. degree from DSCE, Bengaluru, M. Tech., a degree in VLSI and Embedded Systems from SJCE, Mysore, Pursuing a Ph.D. on "Security in IoT at the Physical Layer" in Ponnaiyah Ramajayam Institute of Science and Technology, deemed to be University, Thanjavur, Tamil Nadu, presently working as an Associate Professor in the Department of Computer Science and Engineering in BMSIT &M, Bengaluru. Fields of interest are Computer Networks, Cryptography, microcontrollers, and Embedded systems, DAA, Programming with C and Data structure, Electronic Circuits, Advanced computer architecture, and Python Applications. With 20 years of rich experience in teaching and 4 years of research experience, she has guided final-year BE students and M.Tech students in her carrier. Published 25 research papers in national and international journals and conferences; Scopus indexed and attended several FDPs. She can be contacted at email: bharavi_kumar@bmsit.in.

Smitha Elsa Peter completed her Bachelor of Engineering Degree in ECE from the University of Madras in the year 2001. She completed her Master of Engineering in Applied Electronics from Karunya University in 2006. She completed her Doctoral program at PRIST University in 2016 in the discipline of ECE. She worked at Ponnaiyah Ramajayam Institute of Science and Technology, Deemed to be University, from 2008, Vallam, and Thanjavur from 2008 onwards. She has a vast teaching experience of over 14.6 years. Her research areas include soft computing applications, IOT, and data sciences. She can be contacted at email: smithasishaj@gmail.com