

ARMOR: An anti-counterfeit security Mechanism for Low cost Radio frequency identification systems

Yildiran Yilmaz, Viet-Hoa Do and Basel Halak

School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK.

Abstract—Counterfeited products are costing the global economy hundreds of billions of dollars annually. Radio frequency identification (RFID) technology provides a promising solution for this problem, wherein each product is fitted with a secure tag, which is difficult to forge. However, RFID technology is faced with numerous security threats, for example, if the communication link between the reader and the tag is compromised, then it will be possible for a malicious adversary to obtain the private data stored on the device. Tag cloning attacks have also been demonstrated to be feasible, which severely undermines the capabilities of the RFID technology to protect against counterfeiting. One solution to this problem is the use of authentication protocol; however, existing schemes do not support mutual authentication and are still vulnerable to tag cloning attacks. In this paper, a new security mechanism is proposed, which consists of a lightweight three-flights mutual authentication protocol and an anti-counterfeit tag design. The proposed solution is based on combining the Rabin public-key encryption scheme with physically unclonable functions (PUF) technology. The security of the proposed protocol is systematically analysed and compared with existing schemes. The implementation cost of the proposed security primitives, assuming the 1024-bit public key, is 10139 GEs, which is suitable for low-cost RFID tags. Our results show that the proposed design is up-to 50% more area-efficient compared to systems based on Elliptic Curve Cryptography (ECC).

Index Terms—RFID, Mutual Authentication, Physically Unclonable Function, Public key, Rabin Cryptosystem

1 INTRODUCTION

Emerging security attacks such as tag cloning on RFID systems can severely undermine their perceived advantages in reducing the risk of counterfeit and forgery; this can be especially problematic if forged items are used in safety-critical applications, for example in the case of pharmaceutical products, where a forged medicine may lead to a loss of life. As the purpose of the RFID tag is to provide authentication of a tagged object, RFID tags have many uses in the prevention of counterfeiting of products, such as medicine of some health industries brands or government documents [1]. In the instance of pharmaceutical products, to prevent the counterfeiting of a drug, the manufacturer records the unique identifier of each RFID-tagged drug before it is shipped. Then the manufacturer reads the id of the tag attached to the drug at the point of sale and compares it with the previously recorded genuine identifier, to authenticate the drug.

RFID has improvements over conventional anti-counterfeiting mechanisms, e.g. holograms, barcodes, etc[1]. RFID tagged products can be identified accurately without physical and visual contact. Therefore RFID technology reduces theft and provides tracking and dynamic pricing of products without affecting supply chain efficiency [2]. As the RFID technology is primarily needed in applications such as fraud protection, secure access and anti-

counterfeiting, authentication protocols are the cornerstone in ensuring both the data confidentiality and integrity of RFID systems. Conventional solutions require a great deal of computation and communication resources which might not be possible to implement in resource-constrained devices e.g. RFID tags [3]. While a number of lightweight authentication protocols have been proposed in [4, 5, 6, 7, 8], none of them offers a complete security solution in the context of the key three qualities of lightweight mutual authentication, availability and tag unclonability. Therefore, none of them is truly secure solution in providing fraud protection, secure access and anti-counterfeiting. For example, none of the schemes proposed in [4, 5, 6] protect against physical cloning attacks. The WIPR protocol proposed by Arbit *et al.* [4] uses the shared public key pre-installed in the tag memory and is vulnerable to tag cloning attack and does not provide mutual authentication. On the other hand, Fu *et al.* [5] employs a symmetric encryption to guarantee data confidentiality and integrity, but managing and storing the secret key is a major weakness, and the system could be vulnerable if this key is compromised [9]. There are other approaches which rely on the use of PUF technology, which enhances the resilience of RFID systems against tag cloning attacks [7, 8]. Gope *et al.* [7] proposed a mutual authentication protocol using a Hash function and a PUF. Although it is claimed that the protocol satisfies all common security requirements including availability, this protocol cannot fully resolve the availability issue posed by desynchronisation attack. Chatterjee *et al.* [8] proposed a mutual authentication protocol based on PUF and ECC. However, it still cannot protect the tags' anonymity and their availability and forward security

• The authors are with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK.
Email: yy6e14@soton.ac.uk, doviethoa@doviethoa.com, basel.halak@ecs.soton.ac.uk

Manuscript received MMM dd, yyyy; revised MMM dd, yyyy.

have not been proved. Furthermore, the use of ECC is an expensive solution for low-cost RFIDs [4].

As such, none of the aforementioned protocols can simultaneously provide the three qualities of lightweight mutual authentication, availability and tag unclonability. To this end, this paper reports the following contributions:

- 1) A new lightweight mutual authentication protocol that combines Rabin public-key encryption with PUF technology in order to introduce the ability for the reader to securely transmit the different public keys to the tag in each transaction, and to prevent tag cloning. The security of Rabin cryptosystem [10] relies on the difficulty of large integer factorisation similarly to the RSA scheme. The prominent advantage of Rabin is the much simpler encrypting operation (modular squaring) than RSA and ECC. Therefore, the encryption process is performed on the tag side, while the decryption process, which requires further processing, is performed on the reader side.
- 2) A detailed hardware design of the security system of the RFID tag.
- 3) A systematic evaluation of the proposed protocol and tag hardware design against security attacks namely eavesdropping, tracking, reader impersonation, desynchronization, replay, tag cloning.

The rest of the paper is organised as follows. In Section 2, the background information on Rabin cryptosystem and PUFs is provided. Section 3 and 4 present the proposed protocol and the security analysis of the system. Subsequently, the tag hardware design is described in Section 5, whereas Section 6 evaluates the system performance and design cost. Finally, Section 7 concludes the paper.

2 BACKGROUND

2.1 Rabin Cryptosystem

The Rabin scheme [10] is an asymmetric cryptosystem that relies on the difficulty of the factorisation of large integer numbers. It is equivalent to RSA in term of security and computational complexity. However, it is much simpler to perform the Rabin encryption process compared to RSA while the Rabin decryption requires more computation and resources than RSA[11].

Rabin cryptosystem's private key consists of two large distinct primes p and q . In order to simplify the decryption, the following condition is used:

$$p \equiv q \equiv 3 \pmod{4} \quad (1)$$

The public key referred to as n is the product of the private key pair:

$$n = p \times q \quad (2)$$

Rabin encryption is simply a modular squaring operation, which is much less complex compared to RSA or ECC at the same security level. This calculation is carried out to generate a ciphertext (C) from a plaintext $M \in 0, 1, \dots, n-1$, as follows:

$$C = M^2 \pmod{n} \quad (3)$$

Rabin decryption finds the square root of the ciphertext C modulo n using the Chinese remainder theorem. Under the condition (1) we can calculate the square root of C modulo p, q as follows:

$$\begin{aligned} m_p &= \sqrt{C} = C^{\frac{p+1}{4}} \pmod{p} \\ m_q &= \sqrt{C} = C^{\frac{q+1}{4}} \pmod{q} \end{aligned} \quad (4)$$

Let y_p, y_q be the Bzout's identity of p and q , the four square roots m_0, m_1, m_2, m_3 of C modulo n are then calculated as follows:

$$\begin{aligned} m_0 &= (y_p \times p \times m_q + y_q \times q \times m_p) \pmod{n} \\ m_1 &= n - m_0 \\ m_2 &= (y_p \times p \times m_q - y_q \times q \times m_p) \pmod{n} \\ m_3 &= n - m_2 \end{aligned} \quad (5)$$

The plaintext M will be one of the four square roots. Additional information, such as, a known part of the plaintext, is required to choose the correct one.

Shamir [12] proposed a randomised variant of Rabin encryption which avoided the complex modular operation. If R is a random number with a bit width longer than the key size, the ciphertext is computed as follows:

$$C = M^2 + R \times n \quad (6)$$

WIPR [4] is an implementation of Rabin encryption specifically designed and highly optimised for ultralightweight passive RFID tags using Shamir's randomized multiplication. For 1024-bit Rabin encryption, the data path area is 4,184 GEs and average power consumption is $11\mu W$. BlueJay hybrid cryptosystem [13] is also a promising implementation of Rabin scheme based on PASSERINE encryption mechanism [14]. For 1024-bit Rabin encryption, the hardware size is under 3000 GEs.

2.2 Physically Unclonable Functions (PUF)

PUF is an integrated circuit which produces different outputs when it is implemented on different chips [15]. Manufacturing variability which causes the variation in the property of circuits is inevitable and uncontrollable, which serves PUF unclonability property. PUF has been used for key generation and authentication [7, 8, 16, 17] as well as anti-counterfeiting, IP protection and licensing [18, 19].

PUFs can be categorised into delay-based PUFs and memory-based PUFs. Delay-based PUFs utilise the differences in the propagation delay to generate the response. The most common types of delay-based PUF are arbiter PUFs (A-PUFs) and ring oscillator PUFs (RO-PUFs). A-PUFs directly extract the propagation delay of the two identical paths [20]. Memory-based PUFs use the data stored in memory under an unstable condition such as power-up SRAM [21] and decay DRAM [22]. Unlike delay-based PUFs, memory-based PUFs are only available in a specific condition (e.g. boot time) and may require special circuits (e.g. voltage control).

TABLE 1: Notations and variables used in this paper

Notation	Description
$a b$	Concatenation of a and b
$\mathcal{H}(x)$	Hash function
$PUF(x)$	PUF response of challenge x
p, q	Rabin private keys
n, s_n	Rabin public key and its signature
uid	Unique ID
tid, s_{tid}	Temporary ID and its signature
R_t	Random number generated by the tag
H_r, H_t	Message hash
k	Transaction number
M	Plain message
C	Encrypted message

3 METHODOLOGY

This section overviews the proposed solution and presents the proposed authentication protocol. The notations and variables used in this paper are summarised in Table 1.

3.1 Overview of the Proposed Solution

Security requirements mentioned in [9] for an RFID system, such as mutual authentication, confidentiality, anonymity, availability, forward security, tag unclonability are utmost prominent in building security protocol for RFID systems. When meeting these requirements, the use of chip area should be considered for the security protocol to be applicable to RFID tags. Therefore, our proposed solution combines PUF technology and Rabin encryption to introduce a new security protocol that satisfies the aforementioned security requirements taking into consideration of area consumption.

In cryptography term, Rabin cryptosystem [10] is defined as a method to encrypt and decrypt the data. In our protocol, Rabin is used in a way that the confidential data ($uid, s_{tid_{new}}, s_{n_{new}}$) are encrypted by the tag then decrypted by the reader because Rabin encryption requires considerably fewer computations than other asymmetric encryption algorithms (e.g. RSA, ECC)[11]. Another cryptographic primitive, PUF is employed to generate the tag-specific secrets to be encrypted by the Rabin. Consequently, Rabin and PUF cryptographic primitives are used to preserve forward security, data confidentiality, anonymity and to prevent attacks, e.g. eavesdropping, modification and tag cloning.

3.1.1 System Model

In the system model, RFID tags are mutually authenticated with the reader. The tags can also send its data to the reader using public key encryption. The tags are resource-constrained but the readers are considered not to have any major limitation in terms of design cost and power consumption. Furthermore, this paper focuses on the tags and their communications with the readers, therefore, the tags consider all other entities of the systems in reality such as the central server, database, etc. as part of the readers.

3.1.2 Threat Assumptions

All the infrastructure behind the readers are considered to be trusted and have the abundant resources to work in a secure manner as shown in [23, 24]. **The adversary can**

impersonate the reader by using a malicious reader but it cannot access the tag database in the trusted system.

The communications between the tags and readers can be accessed and tampered by the adversary, including listening to all messages, jamming the air interface and modifying any communication.

The adversary has full knowledge about the protocol, all the security primitives and algorithms used by the tag. It is assumed that the PUF is designed with tamper-resistant in mind because the effort to probe it including the physical attacks to the internal PUF changes the PUF itself or destroys it. However, the adversary can apply a random input to the tag interface to monitor output.

3.1.3 Design Goals

This protocol is developed based on the WIPR design [4]. It attempts to fulfill the following requirements:

- Mutual authentication: both the reader and the tag must be able to authenticate each other. In the WIPR design, only the reader authenticates the tag.
- Public key transmission: different public and private keys are used in each transaction. The proposed protocol must provide a secure method for the reader to transmit the public key and guarantee its integrity. The tag must be able to determine the authenticity of the public key before using it.
- Unclonability: the tag unique ID and its behaviour cannot be cloned even when the adversary has full knowledge about the tag design.

3.2 Proposed Authentication Protocol

Assume that before the tag is released and in use, the temporary ID (TID) random seed tid has been generated randomly and sent to the tag. The system stores following authentication information of the first K transactions in the database for future use:

- The tag unique identifier uid
- Authentication information of the first K transactions:
 - Transaction number $k = 1, 2, \dots$
 - The TID and its signature $s_{tid} = PUF(tid)$
 - The private key (p, q) and the signature $s_n = PUF(\mathcal{H}(n||tid))$ of the corresponding public key $n = p \times q$

The authentication protocol is represented in Fig. 1. It consists of the following steps:

Phase 1 – Identification: the reader identifies the tag using TID in the following two steps:

- 1) The tag generates a random number R_t . The hash value H_t of the random number R_t and the signature s_{tid} of the current tid is transmitted to the reader:

$$H_t = \mathcal{H}(R_t||s_{tid}) \quad (7)$$

- 2) The reader searches the database such that $\mathcal{H}(R_t||s_{tid}) = H_t$ (R_t, H_t are from the tag, s_{tid} is from each database entry). If the query cannot

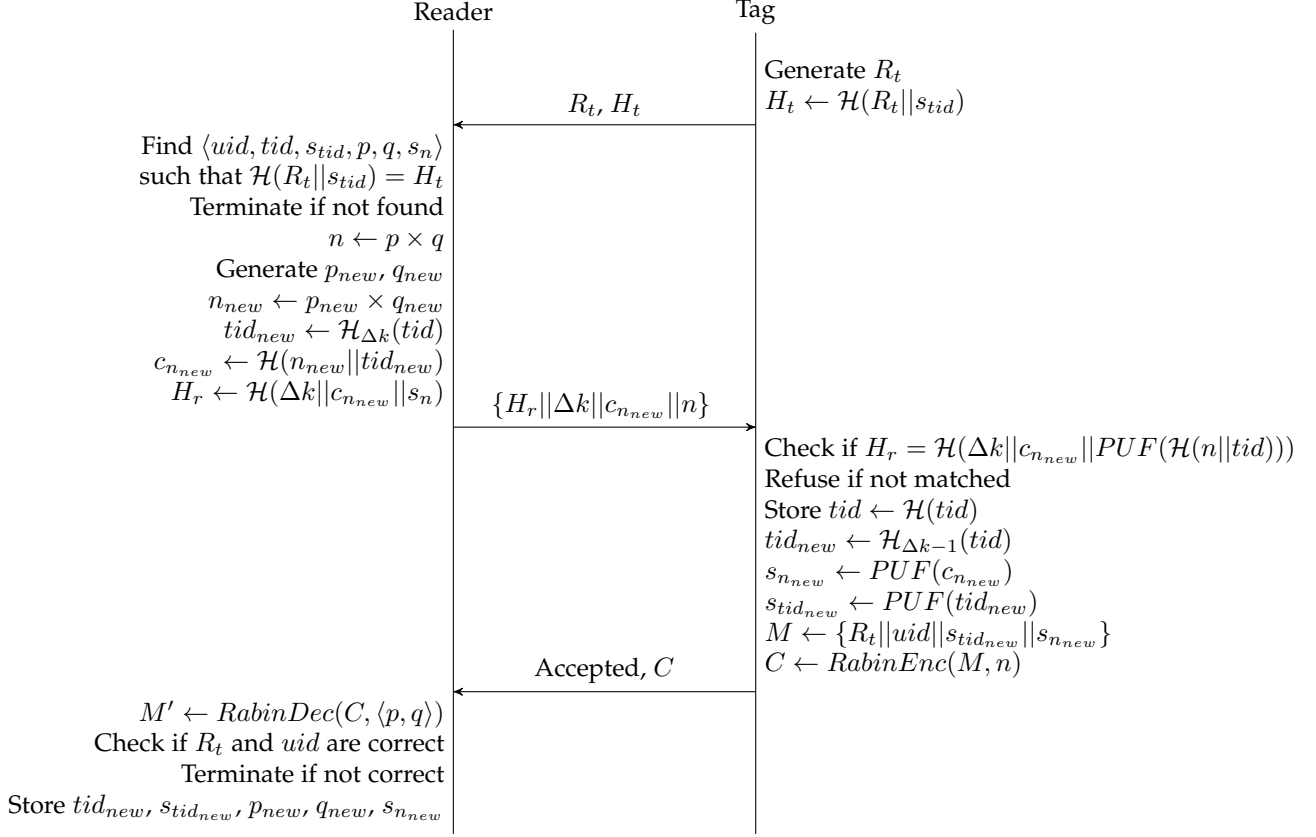


Fig. 1: Proposed mutual authentication protocol

be found, the authentication process is terminated. This identification is carried out based on the PUF response s_{tid} ; consequently, reader distinguishes the PUFs in which used by the tag. If the query is found, the reader uses the tag information to perform authentication in Phase 2. The tag database record contains the following information:

- The unique identifier uid
- The transaction number k
- The current TID tid and its signature s_{tid}
- The current private key (p, q) and the signature s_n of the corresponding public key

Phase 2 – Authentication: the reader and the tag authenticate each other using the shared secrets.

- 1) Let k_{oldest} and k_{newest} be the oldest and the newest transaction information of the same tag in the database. By comparing the transaction numbers k and k_{oldest} , the reader knows the number of transactions the tag has drifted by from the previous successful one is $(k - k_{oldest})$. Depending on the system-specific security policy, it could then decide whether to proceed with the current authentication or perform other emergency actions such as activate the alarm. If the security policy requires the system to be non-tolerant to out-of-sync situations, our protocol can detect an out-of-sync state by checking whether $k > k_{oldest}$ or not. The tag is supposed to be in the transaction, which $k = k_{oldest}$, if there is

no desynchronisation attack. But if an attack desynchronised the tag and the reader, the tag may drift to transaction $k > k_{oldest}$.

If the authentication process continues, the reader prepares to ask the tag for the authentication information of the Δk -next transaction where $\Delta k = k_{newest} - k + 1$. The reader expects the tag to send back the authentication information for the $k + \Delta k$ -th = $k_{newest} + 1$ -th transaction.

The reader computes the public key n from the private key (p, q) for this transaction:

$$n = p \times q \quad (8)$$

The reader then generates a new Rabin private key (p_{new}, q_{new}) and computes the corresponding public key n_{new} by following the rule in [10] such that p_{new} and q_{new} are two large distinct primes that satisfy the condition in Equation (1).

$$n_{new} = p_{new} \times q_{new} \quad (9)$$

The TID and PUF challenge for the Δk -next transaction are computed as follows:

$$tid_{new} \leftarrow \mathcal{H}_{\Delta k}(tid) \quad (10)$$

$$c_{n_{new}} \leftarrow \mathcal{H}(n_{new} || tid_{new}) \quad (11)$$

Note that $\mathcal{H}_{\Delta k}(tid) = \mathcal{H}(\mathcal{H} \dots (tid))$ (Δk times \mathcal{H}).

The reader computes the message signature H_r and transmits n , $c_{n_{new}}$ and H_r to the tag.

$$H_r \leftarrow \mathcal{H}(\Delta k || c_{n_{new}} || s_n) \quad (12)$$

- 2) The tag checks if the message signature H_r from the reader and the value computed by the tag itself match one another. If they are matched, the reader is authentic as it provides the correct s_n , which is secret, and the message itself is intact. Otherwise, the tag transmits a refusal message to the reader and terminates the authentication process.

$$H_r = \mathcal{H}(\Delta k || c_{n_{new}} || PUF(\mathcal{H}(n || tid))) ? \quad (13)$$

Having authenticated the reader, the tag then stores the next TID to the tid memory and computes the secret information for the Δk -next transaction. The state and secret information of the next transaction are computed as follows:

$$\begin{aligned} tid &\leftarrow \mathcal{H}(tid) \\ tid_{new} &\leftarrow \mathcal{H}_{\Delta k-1}(tid) \\ s_{n_{new}} &\leftarrow PUF(c_{n_{new}}) \\ s_{tid_{new}} &\leftarrow PUF(tid_{new}) \end{aligned} \quad (14)$$

The tag encrypts and then transmits its UID uid along with $s_{tid_{new}}$, $s_{n_{new}}$ to the reader using Rabin encryption with the public key n . The randomised variant of Rabin encryption in Equation (6) is used. n_{new} is going to be used in the next authentication process if the current authentication succeeded. n , p and q are going to be discarded if the current authentication is successfully completed.

$$\begin{aligned} M &\leftarrow \{R_t || uid || s_{tid_{new}} || s_{n_{new}}\} \\ C &\leftarrow RabinEnc(M, n) \end{aligned} \quad (15)$$

- 3) The reader decrypts the message C with the private key (p, q) using Equation (5). R_t is the known part of the plaintext which can be used to find the correct square root.

$$M' \leftarrow RabinDec(C, \langle p, q \rangle) \quad (16)$$

If uid sent by the tag is the same as the value stored in the reader, the tag is authentic and fully identified. Otherwise, the authentication process is terminated. **If all the authentication steps are successfully completed, then the reader stores the new secret information tid_{new} , $s_{tid_{new}}$, (p_{new}, q_{new}) and $s_{n_{new}}$ in database for future use and removes the old secret information tid , s_{tid} , (p, q) and s_n . This happens when the third flight is not disrupted and C are decrypted correctly (R_t and uid are correct). But, if the third flight is disrupted after tag computes C , then the reader will not store new secret information and keeps the current secret information in database.**

3.3 Data Retrieval Protocol

By using Rabin encryption it is possible to securely retrieve data from the tag. One of the benefits of this is that it allows the reader to constantly maintain the authentication information of the tag for the next K transactions and recover from desynchronisation attack.

The secret request protocol is depicted in Fig. 2. Each step in the secret request protocol is mostly the same as the authentication protocol.

- The reader generates a new private key $\langle p_{new}, q_{new} \rangle$, the TID tid_{new} of the Δk -next transaction, the PUF challenge $c_{n_{new}}$, and the message checksum H_r and then sends them to the tag.
- The tag checks if the checksum H_r is correct. If so, it computes the secret authentication information $s_{n_{new}}$ and $s_{tid_{new}}$ and encrypts them using the current transaction public key n and subsequently sends them in encrypted form back to the reader.
- The reader decrypts the message using the current transaction private key $\langle p, q \rangle$ and the known part of the plaintext $(uid \oplus H_r)$. If the decryption succeeds, it stores the new secret information tid_{new} , $s_{tid_{new}}$, (p_{new}, q_{new}) and $s_{n_{new}}$ in the database for future use.

3.4 Security Level and Security Primitives Implementation

All the data and the security primitives should be at the same or higher security level to the public key size. For example, for 1024-bit Rabin encryption, the PUF and \mathcal{H} must have at least 80-bit output. It is recommended to refer to the comparative strengths of different cryptosystems proposed by NIST [25].

The protocol prevents the adversary from accessing any PUF CRP (hidden CRP accessibility), therefore using lightweight PUF such as A-PUF is possible [26]. The hash function \mathcal{H} is used to generate the TID, PUF challenges and message checksum.

4 SECURITY ANALYSIS

4.1 Security Attacks

The tag and reader interaction over the insecure environment may face various attacks [27]. These attacks can be listed as follows.

Eavesdropping: In this attack, an attacker who monitors the communication channel, can listen to the message traffic between the tag and the reader and may disrupt the confidentiality of this communication by reading the messages.

Tracking: In this attack, the attacker who intervenes in the communication can obtain the tag's identity information to trace the tag and may violate the anonymity of the tag.

Reader Impersonation: In this attack, the attacker can send messages to the tag by simulating the reader's identity. In this case, if the tag does not perform a reliable mutual authentication, the attacker reader could impersonate readers as if it were an authentic reader in the network and transmit misleading messages to the tag.

Desynchronization: The attacker tries to prevent the continuity of communication by violating the availability of

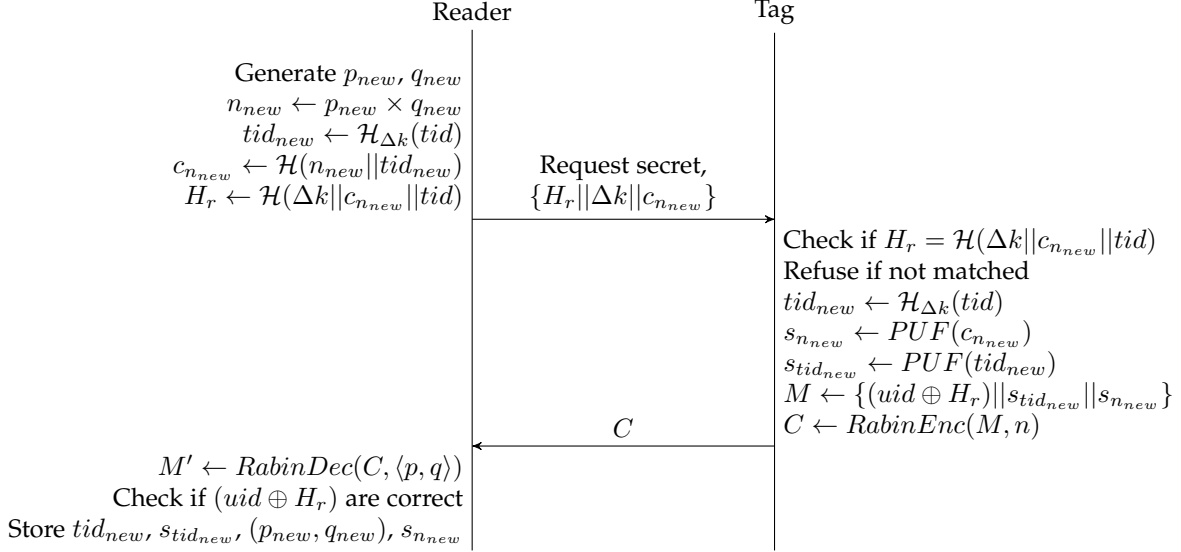


Fig. 2: Secret request protocol

communicating parties. The attack exposes itself in the form of making the data updates between reader and tag desynchronised.

Replay: This attack occurs in the form of malicious and fraudulent repetition or delay of valid data communication.

Tag Cloning: In the cloning attack, an attacker first obtains confidential information from a tag, then clones the tag [2]. Thus, the security of the RFID system can be compromised by copying firmware, software, and secret keys since RFID tags deployed in remote environments are exposed to being caught physically and deceived.

In order to make the RFID system secure against the aforementioned attacks, our protocol aims to satisfy the security requirements in the following section.

4.2 Security Requirements

Several security requirements in [9] for the communication between RFID tags and readers must be satisfied. Each security requirements from 4.2.1 to 4.2.5 will be defined in this subsection and analysed formally with the Scyther tool [28] in Section 4.4. The tag unclonability requirement will be analysed in Section 4.6. The summary of our protocol security analysis is presented in Table 2.

4.2.1 Mutual authentication

In the context of mutual authentication, the tag and the reader must be able to verify each other's authenticity before any sensitive data is exchanged. The proposed protocol satisfies mutual authentication in the following manner.

The tag first verifies the reader's authenticity by comparing the message signature H_r sent by the reader and the signature computed by the tag itself (see Eqn. 12). Only the tag can generate the correct signature of the public key s_n , because of the PUF, and the tag only sends s_n to the trusted reader, therefore only the trusted reader can generate the correct message signature H_r .

The reader then verifies the tag's authenticity by looking up the TID and the unique identity uid sent by the tag. Only

TABLE 2: Security analysis of the ARMOR protocol and other recent work

Protocols	[4]	[7]	[8]	ARMOR
Security requirements				
Mutual authentication	○	●	●	●
Confidentiality	●	●	○	●
Anonymity	●	●	○	●
Availability	●	● ¹	-	●
Forward security	●	●	-	●
Types of attack				
Eavesdropping attack	●	●	●	●
Replay attack	●	-	●	●
Tracking attack	●	○	○	●
Desynchronisation attack	●	● ¹	●	●
Reader impersonation	●	●	-	●
Tag cloning	○	●	●	●

●: Fully satisfied, ●: Partially satisfied, ○: Unsatisfied
- : Not provided

¹ Mitigated using emergency CRPs

an authentic tag which has the matched TID and PUF CRPs can produce the correct values.

4.2.2 Confidentiality

To ensure confidentiality, all secret and sensitive information must be transmitted securely. Our protocol satisfies confidentiality, since all secret information of the tag including the unique identifier uid , the signature of the next TID $s_{tid_{new}}$ and the public key $s_{n_{new}}$ are always encrypted before they are transmitted.

4.2.3 Anonymity

To ensure anonymity, it is required that the tag cannot be traced by the adversary. Our protocol satisfies anonymity since the adversary cannot know the true UID of the tag because it is always encrypted before transmission. The TID is protected and randomised in each transaction by PUF; therefore there is no correlation between the TID and its true identity.

Other unprotected data such as R_t , $c_{n_{new}}$ and H_r are random by nature. There is no correlation between these

values in one transaction and in any other transaction, even when the tag internal state tid is not changed.

4.2.4 Availability

In an RFID system, attacks on availability could be replay or desynchronisation attacks as discussed in [29]. Therefore, a security protocol needs to provide resistant to such attacks to maintain availability under desired levels inquired by RFID application. The proposed protocol satisfies availability because the protocol can resist replay and desynchronisation attacks. To prevent a replay attack, all the messages are randomised for each tag and each authentication cycles in the proposed protocol. To prevent desynchronisation attack, both the tag and the reader remain synchronised and ready to communicate in the following way. By storing authentication information for K consecutive transactions, the reader can detect if the tag and the reader are desynchronised by comparing k with k_{oldest} (Note that in the case of a desynchronisation attack, $k > k_{oldest}$). After this attack is detected, the reader may query to run the secret request protocol demonstrated in Fig. 2 to retrieve authentic secret information from the tag. Furthermore, after any successful authentication, the reader can ask the tag for more authentication information, refills the safe margin of K consecutive transactions and recovers from the attack completely.

4.2.5 Forward Security

To guarantee forward security, it is required that the prior communications are secure even if secret info is leaked in the present. Our protocol satisfies forward security in the following manner.

The tag random TID tid and its true identifier uid are two important secret information of the tag. However, since the TID is protected by the PUF before it is transmitted, it is not possible for the adversary to identify the tag even when the TID random seed tid has been compromised. The UID (uid) is always transmitted securely using Rabin encryption. Even when the uid is compromised, it is not possible to identify the tag from the encrypted message C because the plaintext M is padded with a random number as shown in Equation (15) and encrypted using the public key provided by the reader.

4.2.6 Tag Unclonability

To ensure tag unclonability, it is essential to secure the data on the tag and the data transmitted on the channel. Tag cloning can be done by physically tampering with the tag and printing properly formatted data to the blank RFID tag or by obtaining the data transmitted in the channel. In our protocol, both the PUF cryptographic primitive which secures data on the tag and the PUF response hiding scheme which secures the transferred data on the channel provide tag unclonability. Rabin encryption in the PUF response hiding scheme shown in Fig. 5 secures the transferred data by encrypting all the sensitive data uid , $s_{n_{new}}$ and $stid_{new}$ which both $s_{n_{new}}$ and $stid_{new}$ are PUF responses.

4.3 Scyther Security Verification Tool

Scyther [28] is an open-source security verification tool which can analyse authentication protocols in a formal

way assuming that encryption methods are fault-free in such way that adversary, unless having an encryption key, cannot decrypt the data. The Scyther tool has own input language similar to the standard C programming language to define the protocols. Our protocol is defined with this language as shown in listing 1. The protocol includes roles of objects involved in, public and private variables, databases, transmitted and received messages between objects, and an order of the exchanged messages. Most importantly, each authentication protocol possesses security properties that must be proved in a systematic manner. In the tool, these properties are called claim events. Unless the value of the variable is not exposed to the adversary, a *secrecy* claim of a variable can remain protected when two reliable parties are in communication with each other. The non-injective synchronisation claim namely *Nisynch* in the Scyther language means that all processes expected to occur in the theoretical definition of the protocol should be performed without error and interruption during the execution of the protocol. This claim shows that the received messages are sent by the sender and the sent messages are received by the receiver. The *commit* claims described in Listing 1 show that all communication partners have agreed on the variable values.

The secrecy of the secret request protocol is verified by Scyther security verification tool. The Scyther verification results shows that tid, uid as well as $s_{n_{new}}, stid_{new}$ will be secret between genuine Tag and Reader and will not be known to the adversary as they are transferred in the channel in an encrypted form.

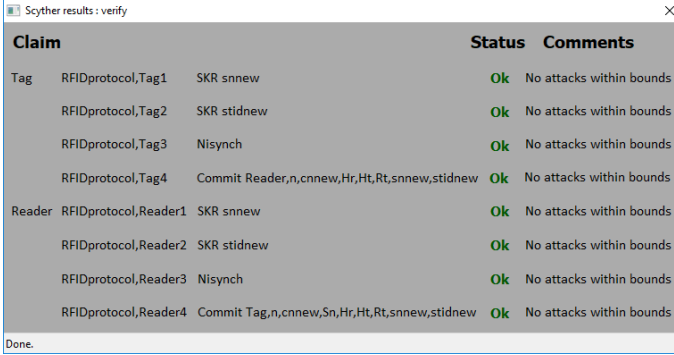
4.4 Defining the ARMOR protocol in Scyther

Listing 1 defines the proposed RFID authentication protocol in the Scyther programming language. There are two roles: the tag's, i.e. *role Tag*, and the reader's, i.e. *role Reader*. The local variables of each role are defined as the nonces $R_t, tid_{new}, stid_{new}, n$ and the challenge $c_{n_{new}}$ and the response $s_{n_{new}}$. The tag and the reader each produce a single nonce, R_t and tid_{new} respectively. The challenge $c_{n_{new}}$ is generated by the tag as a nonce and the responses $s_{n_{new}}$ and $stid_{new}$ are generated by the PUF declared as a constant function, which is only known to the tag and the reader.

Each variable is either specified by declaring it inside the role as *fresh* or *var*. The *fresh* declaration, as well as a variable such as nonce, are randomly generated numbers which are local to the role. The values declared as *var* denote an assigned value upon receipt of a message. *Send* and *receive* events are used in order to send and receive messages respectively between roles. In communication events such as *send* and *recv*, the first parameter denotes the identity of the sender and the second indicates the identity of the receiver.

The rest of the parameters specify the content of the message containing variables. It is required that every variable in the *send* event is given a value. The values from the contents of the *send* event are also assigned to the variables in the *recv* event. The *match* event is used for pattern matching in a way that it assigns the value of the second parameter to the first parameter. Thus, via the *match* event, the output of the PUF function which is the response

computed on the secret key shared between the tag and the reader, i.e. $k(\text{Tag}, \text{Reader})$, and on the challenge $c_{n_{new}}$. The *macro* event defines abbreviations for a particular function to simplify the protocol specification. While the *claim* events define the intended security properties as previously mentioned in Table 2. The tool automatically verifies the security properties described in claim events as shown in Fig. 3.



Claim	Status	Comments
Tag RFIDprotocol,Tag1 SKR snnew	Ok	No attacks within bounds
RFIDprotocol,Tag2 SKR stidnew	Ok	No attacks within bounds
RFIDprotocol,Tag3 Nisynch	Ok	No attacks within bounds
RFIDprotocol,Tag4 Commit Reader,n,cnnew,Hr,Ht,Rt,snnew,stidnew	Ok	No attacks within bounds
Reader RFIDprotocol,Reader1 SKR snnew	Ok	No attacks within bounds
RFIDprotocol,Reader2 SKR stidnew	Ok	No attacks within bounds
RFIDprotocol,Reader3 Nisynch	Ok	No attacks within bounds
RFIDprotocol,Reader4 Commit Tag,n,cnnew,Sn,Hr,Ht,Rt,snnew,stidnew	Ok	No attacks within bounds

Fig. 3: Scyther Verification Results

4.5 Verification of Security Requirements

The results of the security analysis on the proposed authentication protocol using the scyther tool showed that all involved claims are verified as illustrated in Fig. 3. Furthermore, the results of the modelling analysis in Section 4.6 suggest that the protocol provides modelling resistance and tag unclonability.

The security analysis was carried out with the Scyther tool to verify that the authentication protocol is resistant against the following attacks:

An attacker in the *MitM(Man in the middle)* attack is positioned between legitimate parties controlling the authentication protocol. In general, the attacker could compromise an authentication protocol security by exploiting the possibility of changing the content of messages between honest parties seized in the middle. As proven by scyther that the *Nisynch*, *commit* and *SKR(secret)* claims are satisfied, the attacker cannot break the mutual authentication with this kind of attack in the proposed protocol. Therefore, tag and reader mutually have an agreement over the exchanged messages with regard to their order and content.

An *eavesdropping* attack is not possible due to the fact that all unprotected communications are either random or public data and all sensitive data is encrypted. Furthermore, maintaining that the response $s_{n_{new}}$ and private key q are held secret between the tag and the reader ensures that the content of exchanged messages cannot be decrypted.

A *replay* attack is not possible. The signature of the tag's temporary ID s_{tid} , the message signature H_t and the encrypted message C are all randomised for each tag and each transaction. Any part of any communication cannot be partially reused.

A *tracking* attack is also not possible since all data transmitted is either encrypted (C) or random ($R_t, H_t, H_r, c_{n_{new}}$). In addition, private and public keys are dynamic, (i.e. new

Listing 1: Protocol definition in Scyther

```

const PUF: Function;
hashfunction H1;
secret uid: Function;
// sk(X) denotes the long-term private key of Reader
// pk(X) denotes the corresponding public key

protocol RFIDprotocol(Tag, Reader)
{
  role Tag
  {
    fresh Rt, tid, stidnew: Nonce;
    var tidnew;
    var n, snnew, cnnew, Hr, Ht: Nonce;
    match(Ht, H1(Rt, PUF(k(Tag, Reader), tid)));
    send_1(Tag, Reader, Rt, Ht);
    rcv_2(Reader, Tag, Hr, cnnew, n);
    match(snnew, PUF(k(Tag, Reader), cnnew));
    macro tidnew = H1(tid);
    match(stidnew, PUF(k(Tag, Reader), tidnew));
    send_3(Tag, Reader,
           {Rt, uid, stidnew, snnew}pk(Reader));
    claim_Tag1(Tag, SKR, snnew);
    claim_Tag2(Tag, SKR, stidnew);
    claim_Tag3(Tag, Nisynch);
    claim_Tag4(Tag, Commit, Reader, n, cnnew,
              Hr, Ht, Rt, snnew, stidnew);
  }
  role Reader
  {
    fresh n, cnnew, Sn: Nonce;
    var Hr, Ht, Rt, snnew, stidnew;
    rcv_1(Tag, Reader, Rt, Ht);
    match(Hr, H1(cnnew, Sn));
    send_2(Reader, Tag, Hr, cnnew, n);
    rcv_3(Tag, Reader,
           {Rt, uid, stidnew, snnew}pk(Reader));
    claim_Reader1(Reader, SKR, snnew);
    claim_Reader2(Reader, SKR, stidnew);
    claim_Reader3(Reader, Nisynch);
    claim_Reader4(Reader, Commit, Tag, n, cnnew,
                 Sn, Hr, Ht, Rt, snnew, stidnew);
  }
}

```

and different keys are generated after each session). Thus, the tag cannot be traced.

A *desynchronisation* attack can be efficiently mitigated by our protocol. Checking Δk during the protocol process not only allows the reader to detect a desynchronisation attack but also allows the reader to retrieve as much secret information as needed in order to constantly maintain the required error margin.

Reader impersonation is not possible because a fake reader cannot identify the tag from s_{tid} . Furthermore, it is not possible for the fake reader to provide a correct signature of the public key s_n because that value depends on both the tag and the transaction.

4.6 Verification of Tag Unclonability

The proposed tag design is enhanced with PUF technology, which makes it highly resistant to physical cloning attacks. In fact, the only way to clone the tag is to clone the PUF. There are a number of security attacks that aims to clone a PUF as shown in [30], where these can be classified into two types, the first is physical cloning attacks, which have been demonstrated to be extremely difficult. The second type is

based on the use of machine learning algorithms, typically referred to as modelling attacks, wherein, the attacker is assumed to be able to collect a large number of challenge/response pairs (CRP) set by listening to the communication channel [31]. Then, using this set, they attempt to obtain the full CRP sets of the PUF by a numerical model trained with machine learning methods (e.g. support vector machine and neural network). A reasonable approach to prevent such an attack is to conceal the relationship between the pairs of inputs and outputs using cryptosystems [32]. For example, Che *et al.* [33] employed a cryptic hash function applied to the input of the PUF. Cryptographic primitives i.e. Hash function, encryption algorithms and XOR function can be used to hide the relationship between input and output. The challenge permutation and substitution techniques proposed in [34] are lightweight alternatives to make modelling attacks more difficult.

As shown in Fig. 5, we have hidden the PUF response as follows: The response $S_{n_{new}}$ and $S_{tid_{new}}$ produced by the PUF was encrypted using Rabin scheme. The encrypted output C is generated by $C \leftarrow RabinEnc(M, n)$. This prevents any challenge-response sets from being collected from the communication channel by an attacker. Thus he/she cannot create a model for the PUF. We assessed the proposed authentication protocol employing ML attacks (support vector machine and neural network). For comparison, we test the resistance of arbiter against ML attacks.

4.6.1 Test Vector Generation and Machine Learning

In the initial stage of the model building analysis, the challenge-response sets that already utilised within the authentication process have been collected. After the collection has been done, model building attacks based on machine learning techniques are then constructed using these sets. In the traditional PUF authentication protocol, the verification is performed by explicitly transferring the input and output of the PUF between the prover and the verifier as described in [30], so the model building attack is a threat to such protocol because the adversary may build a model obtaining enough number of input and output pairs. Regarding the model building analysis of our protocol, 32000 test vector employing the Matlab program is obtained from Arbiter-PUF. Then we produced 32000 pairs of the encrypted output and the input from our scheme in Fig. 5. We evaluated the test results of SVM and NN using Matlab.

4.6.2 Model-building Results

The SVM and ANN machine learning techniques were performed to examine the strength of the arbiter and the response hiding scheme described in Fig. 5. Arbiter PUF was presented in [35]. In this work, the PUF takes the challenge as a 32-bit input to produce an output as a 1-bit response. During the modelling analysis, b_i refers to the bits of the challenge (C) (i.e. $C = b_1 \dots b_k$) and R refers to the output of the PUF being a response. In addition, the term o_i refers to bits of the encrypted output (C), that is, $C = o_1 \dots o_m$. The value of K is generally considered to be the bit length of the PUF. As described in detail in Fig. 5, the value of the encrypted 1024 bit output is generated by the Rabin encryption using the n as a public key. Accordingly, the value of m for the response hiding scheme is 1024. For

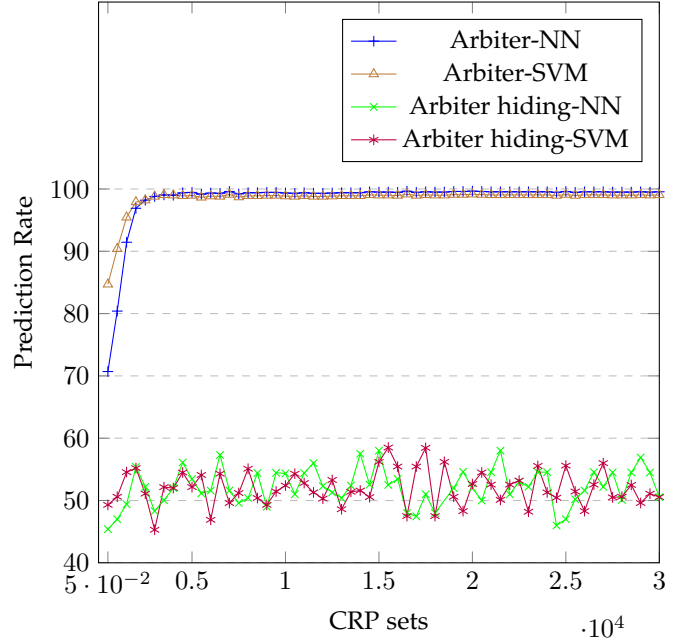


Fig. 4: ML-attack for 40-bit Arbiter PUF and Arbiter-response hiding scheme using SVM and ANN

comparative evaluation, we tested the strength of the 32-bit arbiter facing ML attacks. Fig. 4 shows the prediction rate results of the ML attacks on the arbiter. As shown in Fig. 4, ML techniques have pretty high prediction accuracies of 99.5%(NN) and 98.4%(SVM).

Then the resilience of the response-hiding scheme for the arbiter was tested as well applying the same attacks. Fig. 4 shows the prediction rate results according to both the SVM and NN techniques. Our findings are based on 32000 number of (C_i) challenges and the first 5 bits of the encrypted output o_i i.e. (o_1, o_2, o_3, o_4, o_5). The results from such analysis revealed that the average prediction rate was 52.6% in NN and 51.9% in SVM for all (C_i) and (o_i) bit set. These results highlight that the prediction rate could not be increased by getting further (C_i) and encrypted output (o_i) sets. It is important to note that the same machine learning attacks on the individual output bits of o_i can be employed for analysis of the entire output of the response masking scheme, as in the Arbiter analysis. The finding from the analysis shows that the probability of one bit prediction is $1/2$ for the response masking scheme. The odds of predicting the entire bits of the encrypted output must then be $1/2^m (1/2^{1024})$ in a trial. Consequently, if an attacker attempts to decode a 1024-bit key-coded encrypted output, he or she must perform approximately 2^{1024} computations to obtain the plain text. For RFID tag devices, the 1024-bit security levels support sufficient protection currently. Furthermore, the ML test results show that the difficulty of predicting the entire output of the response hiding scheme is surely increased exponentially by the wrong estimation of one-bit o_i . The results also indicate that the latency of the response guarantees the bit security. Consequently, the introduced response masking technique shows a clear resilience against machine learning attacks. The findings validate the usefulness of encryption algorithms' confusion

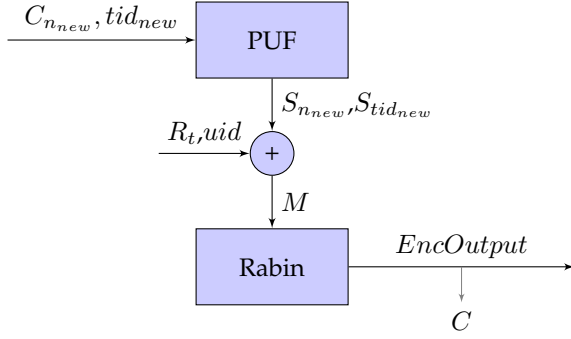


Fig. 5: The response hiding scheme using Rabin algorithm

and diffusion features.

4.7 Comparison with Related Work

The security requirements comparison is provided in Table 2. The proposed protocol is compared with the WIPR design [4], a recent PUF-based authentication protocol proposed in [7] and a recent PUF and ECC-based authentication protocol proposed in [8].

WIPR is a lightweight Rabin scheme authentication protocol and tag hardware design for RFID tags. The protocol is simple and does not support mutual authentication. Furthermore, all the tags share the same public key, and the protocol itself cannot protect the system from tag cloning attack. Our protocol overcomes these drawbacks by using PUF intensively as well as introducing mutual authentication and public key transmission into the protocol.

In term of security, the two protocols reported in [7, 8] are comparable. The main difference is the way the tags identify themselves to the reader, where the tags in [8] transmit their UID in the unprotected channel at the beginning of the transaction while the protocol in [7] uses TID instead. It is obvious that the former approach suffers from tracking attack and violates confidentiality and anonymity requirements, while the later is unable to mitigate tracking attacks against the TID, and also needs to synchronise shared knowledge which introduces potential availability issues.

Our protocol uses TID just like [7] but our solution for the availability problem is complete and more efficient. Note that although Gope *et al.* [7] claimed that their protocol satisfied the availability requirements, they only partially resolve the problem. First, they use a set of emergency CRPs stored in the readers and a set of unique unlinkable pseudo identities stored in the tag in order to recover from a desynchronisation attack. Our protocol does not require any CRP storage in the tag, as it only requires CRPs in the database where resources are considered to be unlimited. Secondly, their protocol does not provide any approach to refill the emergency set after being attacked. Our protocol, however, allows the reader to maintain a number of transactions secret information by obtaining more after any successful authentication.

5 DESIGN AND IMPLEMENTATION

The design of the tag is presented in this section. Because this protocol is developed based on the WIPR design (which

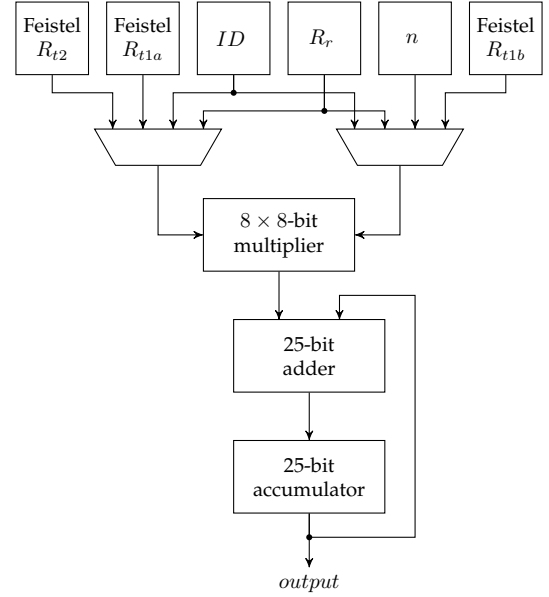


Fig. 6: Data path for WIPR [4] Rabin encryption

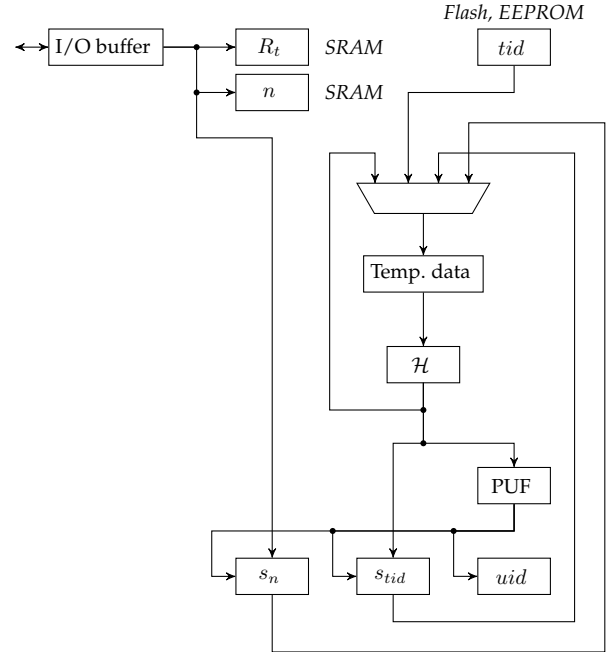


Fig. 7: Extended data path for the proposed protocol

was proposed in [4]), the Rabin encryption is implemented in the same way. The data path of the WIPR encryption is presented in Fig. 6.

In order for the tag to perform the protocol, some additional registers and security primitives are added. The data path of the extended circuit is presented in Fig. 7.

- Memories and registers
 - Public key memory n : 1024-bit single-port SRAM
 - Random value R_r : 80-bit dual-port SRAM
 - Temporary ID tid : 80-bit non-volatile memory (NAND flash, EEPROM)

- s_n, s_{tid}, uid and temporary register for \mathcal{H} : 80-bit registers
- Security primitives
 - Hash: Hash-One
 - PUF: 80-bit Arbiter PUF

5.1 Hash Function \mathcal{H}

The hash function is used to generate the temporary ID, challenge sequence for the PUF and message signature H_t, H_r . \mathcal{H} is not necessarily to be cryptographically strong because it is not directly used for security purpose. Therefore, a simple and area efficient design is preferable. In our reference design, the lightweight cryptographic hash function proposed in [36] is used.

5.2 PUF

In our protocol, an A-PUF [20] is implemented. The A-PUF takes a 40-bit challenge and generates 1-bit response. Therefore in order to generate an 80-bit output, the PUF is fed with a sequence of challenges generated by the hash function \mathcal{H} .

6 EVALUATION

In this section, the statistical characteristics of the authentication protocol and the tag implementation cost are discussed.

6.1 Statistical Characteristics

There are two crucial statistical metrics for authentication protocol: False Acceptance Rate (FAR) and False Rejection Rate (FRR).

- FAR is the probability of the authentication process completing successfully although either the tag or the reader is not authentic.
- FRR is the probability of the authentication process terminating unsuccessfully although both the tag and the reader are authentic.

In practice, the goal is to minimise both FAR and FRR. Since FAR and FRR are both monotonic functions of threshold value with the opposite monotonicity, they reach the optimum balanced value when they are equal. That value is called the Equal Error Rate (ERR).

6.1.1 PUF Performance Metrics

The reliability of the authentication process depends closely on the statistical characteristics of the PUF. Two important performance metrics of PUFs are uniqueness and reliability.

Uniqueness is the ability of PUFs in different devices to generate a unique response for the same challenge. It is the average inter-chip Hamming distance HD_{inter} of the n -bit responses $R_i(n)$ and $R_j(n)$ generated by two different chips i and j respectively. Ideally, the inter-chip Hamming distance of a PUF is 50% [30].

Reliability is the ability of PUFs in the same device to generate the same response for the same challenge. It is the average intra-chip Hamming distance HD_{intra} of the n -bit responses $R_i(n)$ and $R'_i(n)$ generated by the chip i in different conditions. Ideally, the intra-chip Hamming distance of a PUF is 0% [30].

6.1.2 Performance of Single PUF Authentication

Let t be the acceptable number of error bits in the PUF response. Assume that HD_{inter} and HD_{intra} of PUFs follow binomial distribution with the binomial probability estimators of p_{inter} and p_{intra} respectively.

Let $\bar{A}(t)$ and $\bar{R}(t)$ respectively be the FAR and FRR of an n -bit response PUF with the error rate threshold t . $\bar{A}(t)$ and $\bar{R}(t)$ can be calculated as follows [37]:

$$\bar{A}(t) = \sum_{i=0}^t \binom{i}{n} p_{inter}^i (1 - p_{inter})^{n-i} \quad (17)$$

$$\bar{R}(t) = 1 - \sum_{i=0}^t \binom{i}{n} p_{intra}^i (1 - p_{intra})^{n-i} \quad (18)$$

It is obvious that $\bar{A}(t)$ is a monotonically increasing function and $\bar{R}(t)$ is a monotonically decreasing function. The stricter the acceptance threshold t is, the harder it is to accept a wrong tag and the more likely it is to reject the right tag.

6.1.3 Performance of the Authentication Protocol

The successful authentication requires a matching temporary ID s_{tid} , a matching public key signature s_n and a matching unique ID uid which are all PUF responses. However, the protocol accepts the collision of s_{tid} and overcomes such issue by trying all the tags which have a matched s_{tid} . Therefore s_{tid} does not contribute to FAR.

FAR of the authentication protocol is calculated as follows:

$$FAR(t) = \bar{A}^2 \quad (19)$$

An authentication could be incorrectly terminated at either the s_{tid} , s_n or uid checking steps. These values are all generated by the PUF, and therefore the probabilities of each case are \bar{R} , $(1 - \bar{R})\bar{R}$ and $(1 - \bar{R})^2\bar{R}$ respectively.

FRR of the authentication protocol is calculated as follows:

$$FRR(t) = \bar{R} + (1 - \bar{R}) [\bar{R} + (1 - \bar{R})\bar{R}] \quad (20)$$

The FAR and FRR of the authentication protocol have the same monotonicity as the FAR and FRR of the PUF [30]. The goal is to choose the threshold value t_{ERR} such that both FAR and FRR are minimum.

$$t_{ERR} = \underset{t}{\operatorname{argmin}} [\max(FAR(t), FRR(t))] \quad (21)$$

in this case, the equal error rate is

$$ERR = \max(FAR(t), FRR(t)) \quad (22)$$

The FAR and FRR of the Arbiter PUF and the authentication protocol using that PUF are estimated based on the above equations (19) and (20). If the PUF has $p_{intra} = 15\%$ and $p_{inter} = 48.2\%$, the threshold value for the proposed protocol t could be chosen in the range of 26 to 29. This means that if the threshold t_{ERR} is selected within the specified range, FAR and FRR of ARMOR protocol are eliminated effectively.

TABLE 3: Total area of the 1024-bit tag and areas of the constituent important components

Component	Area (GEs)	Area (%)
I/O buffer 8-bit register	48	0.5
s_n, s_{tid}, uid , PRNG state 80-bit registers	2188	21.6
R_{t1a}, R_{t1b}, R_{t2} 80-bit registers	1440	14.2
1024-bit key SRAM	1536	15.1
80-bit R_r SRAM	120	1.2
A-PUF 80-bit	326	3.2
Hash function	1006	9.9
Feistel logic	649	6.4
25-bit adder	113	1.1
8×8 -bit multiplier	394	3.9
25-bit accumulator	150	1.5
Others	2169	21.4
Total area	10139	100

TABLE 4: Comparison between the resources for the tag required by our protocol and other protocols

Protocols	[4]	[7]	[8]	Proposed
Components				
TRNG	●	●	●	●
Hash		●	●	●
PRNG	● ⁽¹⁾	●		●
PUF		●	●	●
ECC			●	
Rabin encryption	●			●
Memories				
Non-volatile (bits)	1024	$128 + 64 \times k^{(2)}$	– ⁽³⁾	80
Registers (bits)	368	576	– ⁽³⁾	568
SRAMs (bits)	0	0	– ⁽³⁾	1104

●: Required ○: Not required

⁽¹⁾ Assume that PRNG is equivalent to the one-way function

⁽²⁾ k is the number of emergency unlinkable pseudo identities

⁽³⁾ Not provided

6.2 Cost Analysis

6.2.1 Area of the Tag

The tag design is synthesised using CMOS $0.35\mu\text{m}$ technology. The area of the 1024-bit tag design and those of some important components are estimated in gate equivalents (GEs) and presented in Table 3.

The total tag design is 10139 GEs which is acceptable for lightweight RFID. Most of the area in the tag is occupied by registers and memories (over 60%) while a considerable amount of area is occupied by multiplexers and control circuits (over 20%). Security primitives, namely PUF and hash function occupies relatively small area, as is indicated by the data in Table 3.

6.2.2 Comparison with Other Authentication Protocols

The same authentication protocols as those discussed in Section 4.7 will be compared in term of design cost to the ARMOR protocol. Each study provides the design cost information in a different way, which makes it much harder to have an accurate total area unless all tag designs are reimplemented entirely. Furthermore, each design may use different implementation for the same security function, such as TRNG, hash function, etc. A summary of the required security primitives and amount of memories of various protocols considered here is shown in Table 4.

In order to have a brief view of the design cost, the area of the tags in each protocol will be estimated based on the following assumptions:

TABLE 5: Area of the security primitive hardware implementations

Type	Name	Security strength	Area (GEs)
[38] TRNG	TRNG	–	72
[36] Hash function	Hash-One	80	1006 ⁽¹⁾
[39] PRNG	xorshift+	80	383
[20] PUF	A-PUF 80-bit	–	326
[40] Encryption	ECC 163-bit	80 ⁽²⁾	12145
[4] Encryption	Rabin 1024-bit	80 ⁽²⁾	1733 ⁽³⁾

⁽¹⁾ Serial variant (smaller area, more cycles)

⁽²⁾ Comparable strength recommended by NIST [25]

⁽³⁾ Excluding payload memory and one way function (counted later as part of the system)

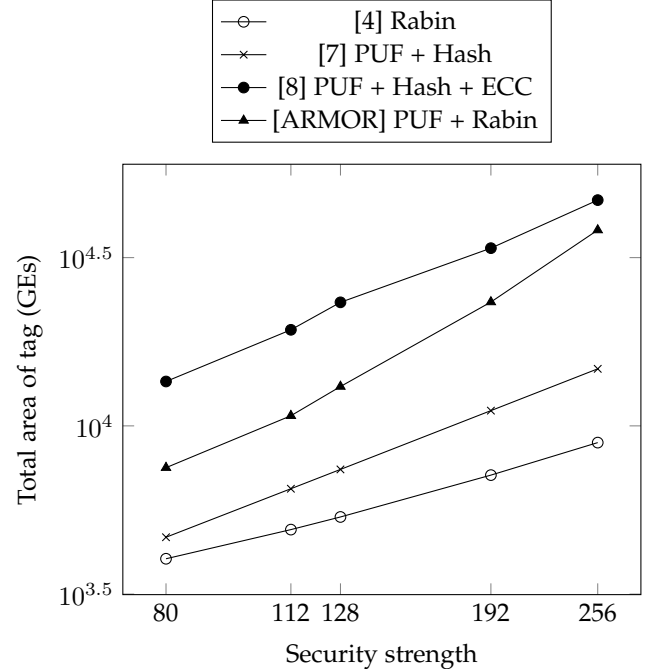


Fig. 8: The estimated total area of the tags in each authentication protocol for different security strengths

- All protocols are implemented using the same security primitives provided in Table 5. Note that the area of the Rabin encryption excludes all memories (which are counted separately, see Table 4) and the one-way function (which is considered equivalent to PRNG).
- The memories bits and the areas of all security primitives except TRNG and Rabin encryption are proportional to their security strength/state bits/challenge bits/key length.

The tag area is estimated for different security strengths from 80 to 256. The key lengths of ECC and Rabin encryption equivalent to each security strength are chosen based on the NIST recommendation [25]. The area of non-volatile memories is considered as negligible, whereas the area of registers and SRAMs are approximately 6 and 1.5 GE/bit respectively. The result is shown in Fig. 8.

The estimated area shows that:

- Our protocol requires more tag area than the protocols based on only Rabin encryption [4] and on

PUF and hash function [7] due to the large memory required for public key and the Rabin encryption block.

- Our protocol requires a smaller area for the tag (20% to 50%) than the protocol based on PUF and ECC which contains the highly complex ECC block. The larger key length of Rabin compared to ECC makes the tag area of our protocol growing faster than the ECC-based design. However, our protocol is still 20% more area efficient at the highest security strength (Rabin key length of 15360 bits).

7 CONCLUSION

The use of RFID technology is a very promising solution to mitigate the risk of counterfeiting. To achieve this, RFID systems need to be resilient against known security threats including tag cloning attacks. Building a secure RFID system is a challenging task due to their limited computing resources. This paper has proposed a lightweight authentication protocol along with secure hardware design for the tag. Our solution combines Rabin public key cryptosystem and PUF technology to perform mutual authentication and public key transmission. The security of the proposed scheme has been systematically analysed and proved against known attacks, including, man in the middle, eavesdropping, replay, tracking, reader impersonation, desynchronisation. In addition, the resilience of the proposed tag design against modelling attacks have been experimentally demonstrated. The evaluation results show that our proposed protocol is up to 50% more area-efficient compared with Elliptic-curve based schemes.

REFERENCES

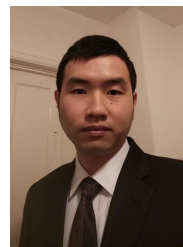
- [1] P. Kitsos and Y. Zhang, *Book of RFID Security*, P. Kitsos and Y. Zhang, Eds. Boston, MA: Springer US, 2008, pp. 1–443, ISBN: 978-0-387-76480-1. DOI: 10.1007/978-0-387-76481-8.
- [2] K. Bu, M. Weng, Y. Zheng, B. Xiao, and X. Liu, “You Can Clone but You Cannot Hide: A Survey of Clone Prevention and Detection for RFID,” *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1682–1700, 2017.
- [3] A. R. Sfar, E. Natalizio, Y. Challal, and Z. Chtourou, “A roadmap for security challenges in the internet of things,” *Digital Communications and Networks*, vol. 4, no. 2, pp. 118–137, 2018.
- [4] A. Arbit, Y. Livne, Y. Oren, and A. Wool, “Implementing public-key cryptography on passive RFID tags is practical,” *International Journal of Information Security*, vol. 14, no. 1, pp. 85–99, 2015.
- [5] L. Fu, X. Shen, L. Zhu, and J. Wang, “A low-cost UHF RFID tag chip with AES cryptography engine,” *Security and Communication Networks*, vol. 7, no. 2, pp. 365–375, 2014.
- [6] D. Wang, Y. Ding, J. Zhang, J. Hu, and H. Tan, “Area-efficient and ultra-low-power architecture of RSA processor for RFID,” *Electronics Letters*, vol. 48, no. 19, pp. 1185–1187, 2012.

- [7] P. Gope, J. Lee, and T. Q. Quek, “Lightweight and Practical Anonymous Authentication Protocol for RFID Systems Using Physically Unclonable Functions,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2831–2843, 2018.
- [8] U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, R. S. Chakraborty, D. Mahata, and M. M. Prabhu, “Building PUF based Authentication and Key Exchange Protocol for IoT without Explicit CRPs in Verifier Database,” *IEEE Transactions on Dependable and Secure Computing*, 2018.
- [9] D. He and S. Zeadally, “An Analysis of RFID Authentication Schemes for Internet of Things in Healthcare Environment Using Elliptic Curve Cryptography,” *IEEE Internet of Things Journal*, vol. 2, no. 1, pp. 72–83, 2015.
- [10] M. O. Rabin, “Digitalized signatures and public-key functions as intractable as factorization,” MIT Laboratory for Computer Science, Tech. Rep., 1979.
- [11] C. Maniavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, “Lightweight Cryptography for Embedded Systems-A Comparative Analysis,” in *Book of Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8247 LNCS, 2014, pp. 333–349.
- [12] A. Shamir, “Memory Efficient Variants of Public-Key Schemes for Smart Card Applications,” in *Workshop on the Theory and Application of Cryptographic Techniques*, Springer, 1994, pp. 445–449.
- [13] M.-J. O. Saarinen, “The BlueJay Ultra-Lightweight Hybrid Cryptosystem,” in *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*, IEEE, 2012, pp. 27–32.
- [14] —, “The PASSERINE Public Key Encryption and Authentication Mechanism,” in *Nordic Conference on Secure IT Systems*, Springer, 2010, pp. 283–288.
- [15] B. Halak, M. Zwolinski, and M. S. Mispan, “Overview of PUF-Based Hardware Security Solutions for the Internet of Things,” in *Circuits and Systems (MWSCAS), 2016 IEEE 59th International Midwest Symposium on*, IEEE, 2016, pp. 1–4.
- [16] G. E. Suh and S. Devadas, “Physical Unclonable Functions for Device Authentication and Secret Key Generation,” in *Proceedings of the 44th Annual Design Automation Conference*, ACM, 2007, pp. 9–14.
- [17] Y. Yilmaz, S. R. Gunn, and B. Halak, “Lightweight puf-based authentication protocol for iot devices,” in *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*, Jul. 2018, pp. 38–43. DOI: 10.1109/IVSW.2018.8494884.
- [18] A. P. Johnson, R. S. Chakraborty, and D. Mukhopadhyay, “A PUF-Enabled Secure Architecture for FPGA-Based IoT Applications,” *IEEE Transactions on Multi-Scale Computing Systems*, vol. 1, no. 2, pp. 110–122, 2015.
- [19] J. Zhang, Y. Lin, Y. Lyu, and G. Qu, “A PUF-FSM Binding Scheme for FPGA IP Protection and Pay-Per-Device Licensing,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1137–1150, 2015.
- [20] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas, “A Technique to Build a Secret Key

- in Integrated Circuits for Identification and Authentication Applications," in *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on*, IEEE, 2004, pp. 176–179.
- [21] B. Narasimham, D. Reed, S. Gupta, E. T. Ogawa, Y. Zhang, and J. Wang, "SRAM PUF Quality and Reliability Comparison for 28 nm Planar vs. 16 nm FinFET CMOS Processes," in *Reliability Physics Symposium (IRPS), 2017 IEEE International*, IEEE, 2017, PM–11.
- [22] A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, B. Skoric, S. Katzenbeisser, and J. Szefer, "Decay-Based DRAM PUFs in Commodity Devices," *IEEE Transactions on Dependable and Secure Computing*, 2018.
- [23] C. C. Tan and J. Wu, "Security in RFID Networks and Communications," in *Wireless Network Security*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 247–267.
- [24] "A Secure Authentication Scheme for RFID Systems," *Procedia Computer Science*, vol. 78, no. December 2015, pp. 100–106, 2016.
- [25] B. Elaine (NIST), "Recommendation for Key Management – Part 1: General," *NIST Special Publication*, pp. 51–54, 2016.
- [26] T. Idriss, H. Idriss, and M. Bayoumi, "A PUF-Based Paradigm for IoT Security," in *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, IEEE, 2016, pp. 700–705.
- [27] M. N. Al Dalaien, S. A. Hoshang, A. Bensefia, and A. R. A. Bathaqili, "Internet of Things (IoT) security and privacy," *Powering the Internet of Things With 5G Networks*, pp. 247–267, 2017.
- [28] C. J. F. Cremers, "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols," in *Computer Aided Verification*, Springer Berlin Heidelberg, 2008, pp. 414–418, ISBN: 978-3-540-70545-1.
- [29] D. Mendez Mena, I. Papapanagiotou, and B. Yang, "Internet of things: Survey on security," *Information Security Journal*, vol. 27, no. 3, pp. 162–182, 2018.
- [30] B. Halak, *Physically Unclonable Functions: From Basic Design Principles to Advanced Hardware Security Applications*. Springer, 2018.
- [31] D. Mukhopadhyay, "PUFs as Promising Tools for Security in Internet of Things," *IEEE Design & Test*, vol. 2356, no. c, pp. 1–1, 2016.
- [32] M. Barbareschi, P. Bagnasco, and A. Mazzeo, "Authenticating IoT Devices with Physically Unclonable Functions Models," *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pp. 563–567, 2015.
- [33] W. Che, F. Saqib, and J. Plusquellic, "PUF-Based Authentication Invited Paper," *IEEE/ACM international conference on Computer-aided design*, pp. 337–344, 2015.
- [34] M. S. Mispan, H. Su, M. Zwolinski, and B. Halak, "Cost-Efficient Design for Modeling Attacks Resistant PUFs," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018*, IEEE, 2018, pp. 467–472.
- [35] B. Halak, Y. Hu, and M. S. Mispan, "Area Efficient Configurable Physical Unclonable Functions for FPGAs Identification," in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, IEEE, 2015, pp. 946–949.
- [36] P. M. Mukundan, S. Manayankath, C. Srinivasan, and M. Sethumadhavan, "Hash-One: A Lightweight Cryptographic Hash Function," *IET Information Security*, vol. 10, no. 5, pp. 225–231, 2016.
- [37] Y. Gao, H. Ma, D. Abbott, and S. Al-Sarawi, "PUF Sensor: Exploiting PUF Unreliability for Secure Wireless Sensing," *IEEE Transactions on Circuits and Systems I*, vol. 64, pp. 2532–2543, 2017.
- [38] P. Z. Wiczorek and K. Golofit, "True Random Number Generator Based on Flip-Flop Resolve Time Instability Boosted by Random Chaotic Source," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1279–1292, 2018.
- [39] S. Vigna, "Further scramblings of Marsaglia's xorshift generators," *Journal of Computational and Applied Mathematics*, vol. 315, pp. 175–181, 2017.
- [40] X. Tan, M. Dong, C. Wu, K. Ota, J. Wang, and D. W. Engels, "An Energy-Efficient ECC Processor of UHF RFID Tag for Banknote Anti-Counterfeiting," *IEEE Access*, vol. 5, pp. 3044–3054, 2017.



Yildiran Yilmaz graduated from the Computer Engineering in Pamukkale University in 2010. He completed his MSc with merit in Cyber Security Department at University of Southampton in the UK in 2015. His MSc project was about enhanced security of any type of File and Text storage. He is currently studying PhD in Electronic and Computer Science at the University of Southampton. His research interests are on the design and implementation of lightweight authentication for the Internet of Things devices, security evaluation of resource-constrained devices and programming and simulations on Contiki operating system.



Viet-Hoa Do is a postgraduate student at the University of Southampton, UK. He received his BSc in Electronics and Telecommunications Engineering at Hanoi University of Science and Technology, Vietnam in 2015. He was an undergraduate research assistant at the Embedded Systems and Reconfigurable Computing Laboratory and a software engineer at Samsung Vietnam Mobile R&D Center.

His research interests include performance, security and reliability optimisation for low cost embedded systems in both software and hardware divisions. He has worked on computer vision and computer graphic for embedded systems, secure hardware design projects.



Basel Halak Dr Basel Halak is the founder and director of the embedded system program at the University of Southampton, he has written over 70 refereed conference and journal papers, and authored two books, including the first textbook on Physically Unclonable Functions. He joined Southampton University in 2011 where he continued pursuing his research on developing reliable and secure systems; he is currently a member of sustainable electronics and materials research group, as well as, the cyber security

group. Dr Halak serves in several technical program committees such as HOST, IEEE IVSW, ICCCA, ICCCS, MTV and EWME. He is an associate editor of IEEE access and an editor of the IET circuit devices and system journal. He is also member of hardware security working group of the World Wide Web Consortium (W3C).