A PUF-based Secure Communication Protocol for IoT

URBI CHATTERJEE, RAJAT SUBHRA CHAKRABORTY and DEBDEEP MUKHOPADHYAY, Indian Institute of Technology Kharagpur

Security features are of paramount importance for IoT, and implementations are challenging given the resource-constrained IoT set-up. We have developed a lightweight identity-based cryptosystem suitable for IoT, to enable secure authentication and message exchange among the devices. Our scheme employs *Physically Unclonable Function* (PUF), to generate the public identity of each device, which is used as the public key for each device for message encryption. We have provided formal proofs of security in the *Session Key security* and *Universally Composable Framework* of the proposed protocol, which demonstrates the resilience of the scheme against passive as well as active attacks. We have demonstrated the set up required for the protocol implementation and shown that the proposed protocol implementation incurs low hardware and software overhead.

Keywords: Internet of Things (IoT) · Identity based Encryption (IBE)· Physically Unclonable Functions (PUFs)

Categories and Subject Descriptors: B.7.1 [**INTEGRATED CIRCUITS**]: VLSI (Very Large Scale Integration); C.2.0 [**COMPUTER-COMMUNICATION NETWORKS**]: Security and Protection; K.5.1 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Security and Protection

General Terms: IoT, Protocol, Security

Additional Key Words and Phrases: Communication protocol, identity-based cryptography, internet-of-things, physically unclonable function, security analysis

ACM Reference Format:
Chatterjee, U., Chakraborty, R. S. and Mukhopadhyay, D. 2015. A PUF-based Secure Communication Protocol for IoT ACM Trans. Embedd. Comput. Syst. V, N, Article A (January YYYY), 29 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

In recent years, the *Internet-of-Things* (IoT) has been foreseen to become an essential landmark in the growth of smart cities and smart homes in the near future. IoT is a set-up in which an unique identifier is assigned to each object, and data exchange between the objects takes place without any human or computer supervision. The IoT is the conjunction of several apparently disparate technologies such as wireless communication, micro-electromechanical systems (MEMS), sensor technologies and the Internet. IoT devices are frequently used in application domains such as environment monitoring, infrastructure management, manufacturing, energy management, medical and health systems, building and home automation and transportation. The IoT nodes such as heart monitoring implants, bio-chip transponders, electric clams in coastal waters, sensor nodes in automobiles, smart thermostats, RFID tags, WiFi connected electronic

Authors' addresses: The authors are with the *Secured Embedded Architecture Laboratory* (SEAL), Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, INDIA – 721302. E-mails: {urbi.chatterjee,rschakraborty,debdeep}@cse.iitkgp.ernet.in.

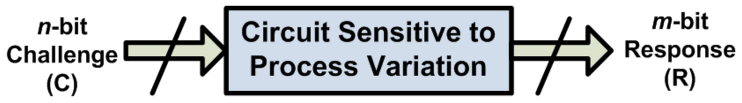


Fig. 1. General working principle of a PUF.

home appliances and smart cards etc., generate large quantities of possibly security-sensitive data. Hence, special methods must be deployed to enable authentication and secure data transmission, and ensure that the IoT nodes are free from the threats of device tampering, information disclosure, privacy breach, denial-of-service, spoofing, elevation of privilege etc. Since the IoT nodes are inherently resource-constrained, an additional challenge in this context is that the implementation of the security measures must be sufficiently lightweight. This often means that traditional cryptographic algorithms and protocols cannot be directly used in IoTs, and novel measures must be adopted.

In recent years, *Physically Unclonable Function* (PUF) [Pappu 2001; LIM 2004] has been introduced as a promising hardware security primitive for different emerging applications. Silicon PUF is a physical entity embodied in a physical structure that is easy to fabricate but practically infeasible to clone, even given the exact manufacturing process that produced it. In addition, the functional mapping between input and output is instance-specific and unpredictable prior to the actual fabrication of the circuit. Rather than embodying a single cryptographic key, PUFs implement a “challenge-response authentication” mechanism. When an electrical stimulus is applied to the structure, it reacts in an unpredictable (but instance-wise repeatable) manner due to the complex interaction of the stimulus with the physical micro-structure of the device. The exact nature of this micro-structure depends on physical factors introduced during manufacturing, which are unpredictable and unclonable. The applied stimulus is considered as the “challenge”, while the reaction generated by the PUF is considered as the “response”. A specific challenge and its corresponding response together form a challenge-response pair (CRP). A given PUF instance’s identity is established by the properties of the micro-structure itself, and its CRP dataset acts as a unique fingerprint for the instance. PUFs have been proposed to be used in IC anti-counterfeiting, device identification and authentication [Majzoobi and Koushanfar 2011], binding hardware to software platforms [Kumar et al. 2008], secure storage of cryptographic secrets [Yu et al. 2011], keyless secure communication [Rührmair 2012], etc. PUF offloads the computational expense of cryptographic algorithms while having relatively low hardware overhead, and is thus a very effective choice for solving secure communication problems in the context of IoTs. Fig. 1 shows the general working principle of a PUF. Since the assessment of a PUF implies a physical measurement, it is very susceptible to circuit noise. Hence to make it reliable and to have full entropy, [Maes et al. 2009] had proposed an error correction circuit with a very low hardware overhead to reduce the fuzziness of the PUF’s responses and make it more robust and reliable.

The desirable features of PUFs are: lightweightedness, unpredictability, unclonability and uniqueness (with respect to each instance). A PUF with a large enough challenge space to make exhaustive enumeration of its CRP set infeasible is termed a *strong PUF*, and are the PUFs of choice in most practical security applications. We would also keep ourselves confined to strong PUFs [Rührmair et al. 2010] in this work. Several strong PUF-based authentication protocols have been proposed in the past, such as *Controlled PUFs* [Gassend et al. 2002], *Noisy PUFs* [Öztürk et al. 2008], *Logically Reconfigurable PUFs* (LRPUF) [Katzenbeisser et al. 2011], *Slender PUFs* [Majzoobi et al. 2012], *Converse PUF-based Authentication* [Koçabas et al. 2012] etc.

However, in [Delvaux et al. 2014], several severe security and practicality issues of the above mentioned protocols were reported. The major concerns with these protocols are: Denial-of-Service (DoS) attack, synchronization problem, replay attack, token/server impersonation, restricted attack model and scalability issues. Similarly, [Avoine et al. 2015] surveys several different ultralightweight authentication protocols designed for RFID authentication, and notes significant threats in the form of typical flaws, e.g. dependency on linear and T-functions, biased output, rotations, insecure message composition, knowledge accumulation by partial leakage of secret and key updating, desynchronization etc. This implies that we cannot adopt any of the above protocols in their original form. To overcome these obstacles, the authors have proposed to use classical challenge-response authentication protocol employing a new ultralightweight cryptographic primitive. Recently, another SRAM PUF based privacy preserving authentication protocol was proposed in [Aysu et al. 2015]. But as the SRAM PUF has been shown to be physically clonable [Helfmeier et al. 2013], we cannot directly adopt this protocol.

In this work we would consider public-key cryptography because of its flexibility and versatility. One important feature of public-key cryptography is its dependency on a public-key infrastructure that is shared among its users. Before the communication, each party must decide its public/private key pair, receive certificates signed by a *Certificate Authority* (CA) after verifying the identity, use them for authentication and encrypt/decrypt the messages. This process can be both laborious and erroneous, and is especially not suitable for IoT applications due to relative lack of resources such as processor speed and memory, and limited interoperability. *Identity-based cryptography* (IBC) can be applied in this situation to reduce these obstacles by requiring no preparation on the part of the receiver. It provides certificate-less public key cryptography, where some unique information about the identity of the user (e.g. a user's email address) is used as its public key. Hence, it eliminates the need for a public key distribution infrastructure. In [Boneh and Franklin 2003], a concept of *Identity based Encryption* (IBE) using *Weil pairing* on elliptic curves was introduced, which provides security against *Chosen-Plaintext Attack* (CPA) and *Chosen-Ciphertext Attack* (CCA). Many lightweight approaches have been proposed for hardware and software implementations for elliptic curve cryptography [Lee et al. 2008; Batina et al. 2006; Batina et al. 2006], and bilinear pairing [Xiong et al. 2010; Yoshitomi et al. 2008; Acar et al. 2011; Grewal et al. 2012; Shirase et al. 2009] respectively, in wireless sensor networks and RFID tags. In the recent past, [Chen 2012] and [Yang et al. 2013] have proposed two IBE based secure communication prototype of for IoT infrastructure.

The other relevant topics of interest are: session key exchange protocols, attack models for such protocols, and their security analysis over an adversary-controlled network. A new notion of *Session Key Security* model and an *Universally Composite Framework* (UCF) were introduced in [Canetti and Krawczyk 2001; Canetti 2001]. Based on this model, different symmetric key exchange schemes using PUFs for resource-constrained devices have been proposed in [Koçabas et al. 2012; Brzuska et al. 2011]. However, as a threat to the above application of PUF, a concept of *malicious PUFs* (namely *PUF re-use* model and *bad PUF* model) was introduced to compromise system security [Ruhrmair and van Dijk 2013], and the same work also discussed techniques to make them secure and robust. In [Ostrovsky et al. 2013], the notion of UCF was modified to make it resistant against malicious PUFs.

1.1. Our contributions and paper organization

In this paper our major contributions are:

- We have presented a complete authentication, key sharing and secure communication architecture where each IoT node has an integrated PUF instance, and the identity of the node is created by the CRP signature of its PUF instance.
- We have considered the *Identity based Encryption* scheme proposed in [Boneh and Franklin 2003], and *Weil Pairing* on elliptic curves which was proved to be secure against CPA and CCA Secure. But The major differences of our scheme with respect to the originally proposed one are:
  - Instead of using public e-mail ID or any publicly known identity string, we have used the response of the PUF on a data node, obtained after applying a pair of challenges, as the public identity of the data node (refer to section 3.3).
  - In our protocol there is no *Public Key Generator* (PKG), as mentioned in the scheme described in [Boneh and Franklin 2003]. Here, the data node which is going to receive the message, generates its own private/public key, and the server is responsible for verifying the public key to prevent any masquerading attack (refer to section 3.3).
  - And the foremost difference is: the original scheme was not protected against *repudiation* attack. With respect to IoT framework, it is very crucial to keep track of which device is sending what data. It can help to find out the device in case of any malicious information damages the system. If a system is secure against repudiation attack, the device which is sending the malicious data, cannot deny its accountability later for the mishap.
- We demonstrate through theoretical analyses that our proposed scheme is resistant against different active and passive attacks in *Session Key Security* model and *Universally Composite Framework*. Our security proofs are based on reduction of the problem of attacking the proposed scheme to the problem of cloning (either physically or mathematically) a particular PUF instance. **Since it is known that: (a) physical cloning of most PUF variants; (b) mathematical cloning of particular PUF variants, or, (c) devising a physical PUF instance or a mathematical model sufficiently similar to a given PUF instance, are difficult problems at the current state-of-the-art, we conclude that the proposed schemes are secure.**
- We have also implemented the major software and hardware components of the protocols and demonstrated that they incur low hardware and software overhead.

The rest of the paper is organized as follows. In Section 2, we provide the necessary background on cryptographic pairing. In Section 3, we present our proposed novel authentication, key exchange and secure communication protocol. The security analysis of the proposed scheme is described in Section 4, along with a detailed comparison of the proposed protocol with existing protocols. The experimental set up and results have been given in Section 5. We conclude the paper with future research directions in Section 6.

2. BACKGROUND: CRYPTOGRAPHIC PAIRING

First we will provide the mathematical theory of Elliptic curve cryptography and then define pairings on elliptic curves as bilinear maps on additive groups on that basis. The definition and the notations are referred from [Mass 2004].

2.1. Elliptic Curves

Let K denote a field and $\overline{K}$ its algebraic closure. We denote K^* for $K \setminus \{0\}$. Consider the homogeneous function over the projective plane $\mathbb{P}^2(\overline{K})$ given by:

$$F(X,Y,Z) = Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3 \tag{1}$$

where $a_1, a_2, a_3, a_4, a_6 \in \overline{K}$. An elliptic curve E is defined to be the set of solutions to $F(X, Y, Z) = 0$ in the projective plane $\mathbb{P}^2(\overline{K})$, where points are equivalent to their multiples, i.e. $(X : Y : Z) \backsim (\lambda X : \lambda Y : \lambda Z)$ for $\lambda \in \overline{K^*}$. There is only one point in E with $Z = 0$ which is termed as the point at infinity $\mathcal{O}$ (denoted by $(0 : 1 : 0)$) . Next, we define a *non-singular curve* where, for all points $P \in E$, the three partial derivatives $\delta F/\delta X$, $\delta F/\delta Y$ and $\delta F/\delta Z$ are not all zero at P.

Additionally, we will use affine coordinates $x = X/Z$ and $y = Y/Z$, and express an elliptic curve by the (*affine*) Weierstrass equation:

$$E\colon y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2}$$

Formally the elliptic curve is the set of points $(x, y) \in \mathbb{P}^2(\overline{K}) = \overline{K} \times \overline{K}$ that satisfy equation 2, along with the point at infinity $\mathcal{O}$. E is said to be defined over K, denoted E/K, if $a_1, a_2, a_3, a_4, a_6 \in K$. Then K is called the definition field and $E(K)$ is denoted by the set of K-rational points, i.e. the points with both coordinates in K, together with $\mathcal{O}$. Now, we will represent $E = E(\overline{K})$.

Two elliptic curves E_1/K, E_2/K are said to be isomorphic over K (denoted as $E_1/K \backsimeq E_2/K$), if there exist $u, r, s, t \in K$, $u \neq 0$, such that the admissible change of variables $(x, y) \longrightarrow (u^2x + r, u^3y + u_2sx + t)$ transforms the equation of E_1 into the equation of E_2. Generally the elliptic curves over a finite field $K = \mathbb{F}_q$ consists of q elements, where $q = p^m$ is a prime power. So the algebraic closure of K is given by $\overline{K} = \cup_{i \geq 1} F_{q^i}$. If the characteristic p of K is greater than 3, then any curve defined over K is isomorphic with a curve of particularly simple form, namely:

$$E\colon y^2 = x^3 + ax + b; a, b \in K \tag{3}$$

There is *tangent-and-chord-method* that creates a additive group using the set of points on an elliptic curve and it has the point at infinity $\mathcal{O}$ as zero element. Let $P = (x_1, y_1), Q = (x_2, y_2) \in E \backslash \{\mathcal{O}\}$ for characteristic > 3. The point $-P$ is given by $(x_1, -y_1)$. Suppose $Q \neq -P$, then $P + Q = (x_3, y_3)$, where

$$x_3 = \lambda^2 - x_1 - x_2 \tag{4}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \tag{5}$$

$$\lambda = \begin{cases} (y_2 - y_1)/(x_2 - x_1), & \text{if } P \neq Q \\ (3x_1^2 + a)/2y_1, & \text{if } P = Q \end{cases} \tag{6}$$

Similarly for $m \in \mathbb{Z}$ and $P \in E$, the scalar multiplication by m is defined as:
$[m]P = P + ... + P$ (m terms) for $m > 0$, $[0]P = O$, and $[-m]P = [m](-P)$ for $m < 0$.

The order of a point $P \in E$ is the smallest positive integer k such that $[k]P = \mathcal{O}$ and the order is said to be infinite in case no such integer exists. If $[n]P = \mathcal{O}$ for $P \in E$, then P is called an n-torsion point. The subgroup of n-torsion points in E is denoted by $E[n]$, so

$$E[n] = \{P \in E : [n]P = \mathcal{O}\}.$$

Moreover, we can write $E(K)[n]$ for the subgroup of n-torsion points in $E(K)$, so

$$E(K)[n] = \{P \in E(K) : [n]P = \mathcal{O}\}.$$

Let E be an elliptic curve defined over $\mathbb{F}_q$. The number of points in $E(\mathbb{F}_q)$, called the order of the elliptic curve, is denoted by $\#E(\mathbb{F}_q)$. The trace of Frobenius or simply trace of a curve is the value t satisfying $\#E(\mathbb{F}_q) = q + 1 - t$. The elliptic curve $E/\mathbb{F}_q$ is said to be supersingular if the characteristic p of $\mathbb{F}_q$ divides t, and non-supersingular otherwise.

2.2. Functions on an Elliptic Curve

Let $K = \mathbb{F}_q$ and let the curve E/K be given by the Weierstrass equation

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{7}$$

We define the function $r \in K[x, y]$ by

$$r(x, y) = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6,$$

then the coordinate ring $K[E]$ of E over K is defined as the integral domain

$$K[E] = K[x, y] = (r),$$

where (r) is the ideal in $K[x, y]$ generated by r. Similarly, $\overline{K}[E]$ is defined as

$$\overline{K}[E] = \overline{K}[x, y] = (r)$$

We can write $l \in \overline{K}[E]$ in canonical form,

$$l(x, y) = v(x) + yw(x), v(x), w(x) \in K[x],$$

by using the relation 2 to knock down exponents of y that are greater than or equal to 2. By taking the field of fractions of $K[E]$, we obtain the function field $K(E)$. Here, two elements f_1/g_1 and $f_2/g_2 \in K(E)$ are identified if $f_1g_2 = f_2g_1$. Similarly for $\overline{K}$, $\overline{K}(E)$ is the field of fractions of $\overline{K}[E]$. An element of $\overline{K}(E)$ is called a rational function.

Definition 2.1. **Zeros and Poles of a Function:** A non-zero rational function $f \in \overline{K}(E)^*$ is said to be defined at a point $P \in E\ \mathcal{O}$ if f can be written as $f = g/h$, for $g, h \in K[E]$, with $h(P) \neq 0$. In that case, the evaluation of f at P, given by $f(P) = g(P)/h(P)$, is well-defined. Further, f is said to have a zero at P if $f(P) = 0$, and f is said to have a pole at P (denoted by $f(P) = \infty$) if f is not defined at P. It is to be noted that a non-zero rational function can only have a finite number of zeros and poles.

Definition 2.2. **Uniformizing Parameter of a Point:** For every point $P \in E$, there exists a rational function u with $u(P) = 0$, such that every non-zero rational function $f \in \overline{K}(E)^*$ and is written as $f = u^d s$, where $s \in \overline{K}(E)$, $s(P) \notin 0, \infty$, for some integer d. This function is called a uniformizing parameter for P.

Definition 2.3. **Multiplicity of Zeros and Poles:** Let u be a uniformizing parameter for $P \in E$. The non-zero rational function $f \in \overline{K}(E)^*$ can be decomposed as $f = u^d s$, where $s \in \overline{K}(E)$ and $s(P) \notin 0, \infty$. Then the order of f at P, denoted as $ord_P(f)$, equals d. If P is a zero of f, then $ord_P(f) > 0$ and the zero is said to have multiplicity $ord_P(f)$. In case P is a pole, then $ord_P(f) < 0$ and the multiplicity of the pole is defined as $-ord_P(f)$. If P is neither a zero nor a pole, then $ord_P(f)$ = 0.

Definition 2.4. **Divisors:** Next we define a divisor D as a formal sum of points

$$D = \sum_{P \in E} n_P(P)$$

where $n_P \in \mathbb{Z}$ and n_P = 0 for all but finitely many $P \in E$. The group of divisors of E, denoted $Div(E)$, is the free abelian group generated by the points of E, where addition is given by

$$\sum_{P \in E} n_P(P) + \sum_{P \in E} m_P(P) = \sum_{P \in E}(n_P + m_P)(P)$$

The support of a divisor $D = \sum_{P \in E} n_P(P) \in Div(E)$ is given by the set of points

$$supp(D) = \{P \in E | n_P \neq 0\}.$$

Further, its degree $deg(D)$ is defined by

$$deg(D) = \sum_{P \in E} n_P.$$

The divisors of degree 0, denoted $Div^0(E)$, form a subgroup of $Div(E)$. For any degree zero divisor $D \in Div^0(E)$, there is a unique point $P \in E$ such that $D \sim (P) - (\mathcal{O})$. In canonical form, it is written as:

$$D = (P) - (\mathcal{O}) + div(f) \tag{8}$$

where $f \in \overline{K}(E)$ is uniquely determined up to a constant multiple.

Definition 2.5. **Principle Divisor:** The divisor of a function f, denoted $div(f)$, as:

$$div(f) = \sum_{P \in E} ord_P(f)(P).$$

Note that $div(f)$ = 0 if and only if f is a constant function. A divisor $D \in Div(E)$ is called principal if $D = div(f)$ for some rational function f. Two divisors $D_1, D_2 \in Div(E)$ are said to be (linearly) equivalent, denoted $D_1 \backsim D_2$, if $D_1 - D_2$ is principal, i.e. we can write:

$$D_1 = D_2 + div(f) \tag{9}$$

for some rational function f.

Definition 2.6. **Picard Group:** The set of all principal divisors is denoted $Prin(E)$. The quotient group

$$Pic(E) = Div(E)/Prin(E)$$

, which represents the set of divisors that are not principal, is called the Picard group or divisor class group. From the fact that

$$div(f_1 f_2) = div(f_1) + div(f_2)$$

for all $f_1, f_2 \in \overline{K}(E)$ follows that $Prin(E)$ forms a subgroup of $Div^0(E)$. Hence, we can define the group called the degree zero part of the Picard group, or divisor class group of E, as follows:

$$Pic^0(E) = Div^0(E)/Prin(E).$$

Finally we will conclude this subsection by the Weil's reciprocity law which states:

THEOREM 2.7. *(Weil's reciprocity law) Let $f, g \in \overline{K}(E)$. Then $f(div(g)) = g(div(f))$*

2.3. Weil Pairing

Definition 2.8. Let $F : E \longrightarrow E$ be a nonconstant rational mapping of smooth curves, $P \in E$ and $u \in K(E)$ is a uniformizing parameter for $F(P)$. Then the ramification index of F at P is defined by

$$e_F(P) = ord_P(u \circ F)$$

By the definition of the order of a point, $e_F(P)$ is independent of the choice of u and that $u \circ F$ has a zero at P, and $e_F(P) \geq 1$.

Definition 2.9. Let $F : E \longrightarrow E$ be a nonconstant rational mapping of smooth curves. The homomorphism $F^* : Div(E) \longrightarrow Div(E)$ is given by

$$F^*((Q)) = \sum_{F(P)=Q} e_F(P).(P)$$

Let $m \geq 2$ be a fixed integer coprime to $p = char(K)$. Suppose $T \in E[m]$, then, the divisor $m(T) - m(\mathcal{O})$ is principal. Let $f \in \overline{K}(E)$ be a function such that $div(f) = m(T) - m(\mathcal{O})$.Therefore, the divisor

$$\begin{aligned}[m]^*(T) - [m]^*(\mathcal{O}) &= \sum_{[m]P=T} e_{[m]}(P).(P) - \sum_{[m]P=\mathcal{O}} e_{[m]}(P).(P) \\ &= \sum_{R\in E[m]} (T' + R) - (R),\ s.t.\ [m]T' = T,\ T' \in E\end{aligned}$$

Since $e_{[m]}$=1, this has degree 0. Since $\#E[m] = m^2$ and $[m^2]T' = \mathcal{O}$,

$$\sum_{R\in E[m]} (T' + R) - (R) = [m^2]T' = \mathcal{O}$$

Therefore, $[m]^*(T) - [m]^*(\mathcal{O})$ is a principle and there is a function $g \in \overline{K}(E)$ with $div(g) = [m]^*(T) - [m]^*(\mathcal{O})$. Now we can define Weil Pairing.

Definition 2.10. The Weil e_m-pairing $e_m : E[m] \times E[m] \longrightarrow \mu_m$ is given by:

$$e_m(S, T) = g(X + S)/g(X)$$

where $X \in E$ is any point such that $g(X + S), g(X) \notin 0, \infty$, and μ_m is the group of m-th roots of unity in K.

THEOREM 2.11. ***(Properties of Weil Pairing):*** *Let $S_1, S_2, S; T_1, T_2, T \in E[m]$. The Weil pairing satisfies the following properties:*

(1) $e_m(S_1 + S_2, T) = e_m(S_1, T)e_m(S_2, T)$ (linearity in the first factor).
(2) $e_m(S, T_1 + T_2) = e_m(S, T_1)e_m(S, T_2)$ (linearity in the second factor).
(3) $e_m(S, S) = 1$ (identity).
(4) $e_m(S, T) = e_m(T, S)^{-1}$ (alternation).
(5) If $e_m(S, T) = 1$ for all $S \in E[m]$, then $T = \mathcal{O}$ (non-degeneracy).

The details of these properties are given in [Silverman 1986].

2.4. Tate Pairing

Similar to the Weil pairing, the Tate pairing is defined in terms of rational functions evaluated in a certain divisor. Let $P \in E(F_{q^k})[m]$, then $m(P) - m(\mathcal{O})$ is a principal divisor. Let $g \in F_{q^k}(E)$ with $div(g) = m(P) - m(\mathcal{O})$. Now let Q be a point representing a coset in $E(F_{q^k})/mE(F_{q^k})$, then we construct a divisor $D \in Div^0(E)$ such that $D \backsim (Q) - (\mathcal{O})$. D is selected such that its support is disjoint from the support of the divisor of g. Next, we can define Tate Pairing as:

Definition 2.12. the Tate pairing is given by:

$$< ., . >: E(F_{q^k})[m] \times E(F_{q^k})/mE(F_{q^k}) \longrightarrow F^*_{q^k}/(F^*_{q^k})^m$$

is given by: $< P, Q >= g(D)$.

The properties of Tate pairing are given below:

(1) $< \mathcal{O}, Q > = 1$ for all $Q \in E(F_{q^k})$ and $< P, Q > \in (F^*_{q^k})^m$ for all $P \in E(F_{q^k})[m]$ and all $Q \in E(F_{q^k})/mE(F_{q^k})$ (well-defined).
(2) For each point $P \in E(F_{q^k})[m]/\mathcal{O}$, there is some point $Q \in E(F_{q^k})$ such that $< P, Q > \notin (F^*_{q^k})^m$ (non-degeneracy).
(3) For all $P_1, P_2, P \in E(F_{q^k})[m]$ and $Q_1, Q_2, Q \in E(F_{q^k})$, we have $< P_1 + P_2, Q > = < P_1, Q >< P_2, Q >$, and $< P, Q_1 + Q_2 > = < P, Q_1 >< P, Q_2 >$ (bilinearity).

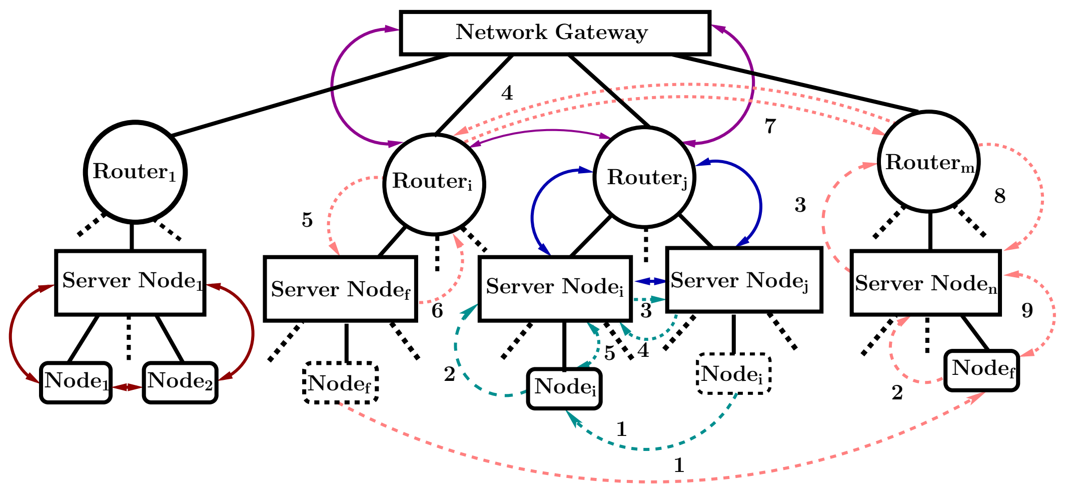
3. PROPOSED AUTHENTICATION, KEY EXCHANGE AND SECURE COMMUNICATION PROTOCOL

In this section, we describe a protocol that provides the functionalities of authentication, key exchange and secure communication, and can be suitably implemented in an IoT infrastructure.

The IoT network consists of several mobile “data nodes” (referred as $Node_1, Node_2, ..., Node_i$ in Fig. 2), which act as sources and recipients of information. A static “server node” (referred as $Server\ Node_1, ..., Server\ Node_n$) is used to coordinate the communication of each node in its read range, and pass the data to a “router” (referred as $Router_1, ..., Router_m$). The router consequently passes the information to the cloud through the “Network Gateway” to be utilized [Murthy and Manoj 2004]. The data nodes are constrained in computational resources, while the server nodes and all the other higher level nodes are relatively resourceful. It is assumed that the server nodes and the data nodes are so distributed that each data node is within the read range of at least one server node at all times. Each data node, server node and router possess a PUF instance which is used to authenticate them to the server node, router and the network gateway, respectively. Note that in our proposed architecture there is no explicit key storage at any of the nodes. We will describe our protocol with respect to the data nodes and the server nodes; however it is to be noted that this mechanism extends to higher level nodes in the infrastructure. The proposed protocol follows the following steps as described below:

- Initially, an **Enrolment Phase** is executed, once for each data node being managed by a given server node. This procedure creates a CRP database for the PUF contained in each node, in the server node corresponding to the data node.
- When two nodes (supervised by the same server node) want to communicate, their common server node first authenticates them, and then helps them to generate their pubic/private key pairs, and then enables secure key sharing. This is termed the **Authentication and Key Sharing Phase.**
- Finally, in the **Secure Communication Phase**, the two nodes send and receive the message securely over the network using the keys, without any intervention of the server.
- If a data node moves to a new location so that it has to be managed by a new server node, the authentication procedure has to be carried out between the data node and the new server node supervising it.

It has been assumed that each data node holds the address of the server node known as “home address” to which it is enrolled in. When it moves to the domain of a new server node, a “handoff mechanism” [Murthy and Manoj 2004] reassigns the node to the new server. To initiate it, the data node sends the authentication request to the new server node along with its home address. If the address is within the range of the router to which the new server belongs, then it randomly chooses two challenges and sends an encrypted message to the old server asking for the responses corresponding to the data node’s CRP database. As these two server are under the same router, it can be assumed that they already have session key pairs or can create a pair by authenticating to the



↔ Key generation and secure communication between Node$_1$ and Node$_2$ are done after the authentication by Server Node$_1$.

↔ Key generation and secure communication between Server Node$_i$ and Server Node$_j$ are done after the authentication by Router$_j$.

↔ Key generation and secure communication between Router$_i$ and Router$_j$ are done after the authentication by Network Gateway.

⇢ 1. Node$_i$ moves from the domain of Server Node$_j$ to Server Node$_i$.
2. Node$_i$ requests for authentication to Server Node$_i$.
3. Server Node$_i$ randomly selects two challenges and requests Server Node$_j$ for the responses from CRP database of Node$_i$.
4. Server Node$_j$ sends the responses to Server Node$_i$.
5. Node$_i$ is authenticated by Server Node$_i$ using the two challenge – response pairs.

⇢ 1. Node$_f$ moves from the domain of Router$_i$ to Router$_m$.
2. Node$_f$ requests for authentication to Server Node$_n$.
3. Server Node$_n$ randomly selects two challenges and forwards the request to Router$_m$.
4. Router$_m$ requests Router$_i$ for the responses from CRP database of Node$_f$.
5. Router$_i$ forwards the request to Server Node$_f$.
6. Server Node$_f$ sends the responses toRouter$_i$.
7. Router$_i$ forwars the responses to Router$_m$.
8. Router$_m$ forwards the responses to Server Node$_n$.
9. Node$_f$ is authenticated by Server Node$_n$ using the two challenge – response pairs.

Fig. 2. The secure communication mechanism in different levels of IoT architecture.

router. Once the old server receives the message, it retrieves the responses from the data node's CRP database and forwards it back to the new server. Consequently, the new server authenticates the data node using these two challenge-response pairs.

However, if the old server resides in the domain of a different router, then the new server forwards the request to its router. It then finds out the router of the old server and sends the message to it (using their public/private key pairs). The router of the old server retrieves the responses from it and again forwards them to the router of the new server. It comes back to the new server later and is used for authenticating the data node by the new server. Once authentication is done, the data node can establish key pairs with any other data node in the new server's domain.

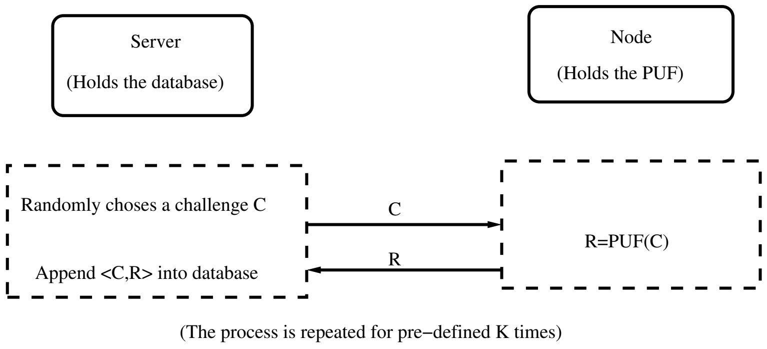

Server
(Holds the database)
Node
(Holds the PUF)
Randomly choses a challenge C
Append <C,R> into database
C
R
R=PUF(C)
(The process is repeated for pre−defined K times)

Fig. 3. The enrolment phase of the proposed protocol.

3.1. Public Mathematical Parameters

Our scheme requires that the communicating parties must agree on several mathematical parameters before initiating communication. For some large prime value p, two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of order p (as mentioned in Section 2) are generated, and an admissible bilinear map ê: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is defined over these two groups. We also need to choose four secure cryptographic hash functions:

- H_1: $\{0,1\}^n \rightarrow \mathbb{G}_1^*$
- H_2: $\mathbb{G}_2 \rightarrow \{0,1\}^n$
- H_3: $\{0,1\}^n \times \{0,1\}^n \rightarrow Z_p^*$, where Z_p^* is a set non-negative integers less than p and co-prime to p.
- H_4: $\{0,1\}^n \rightarrow \{0,1\}^n$

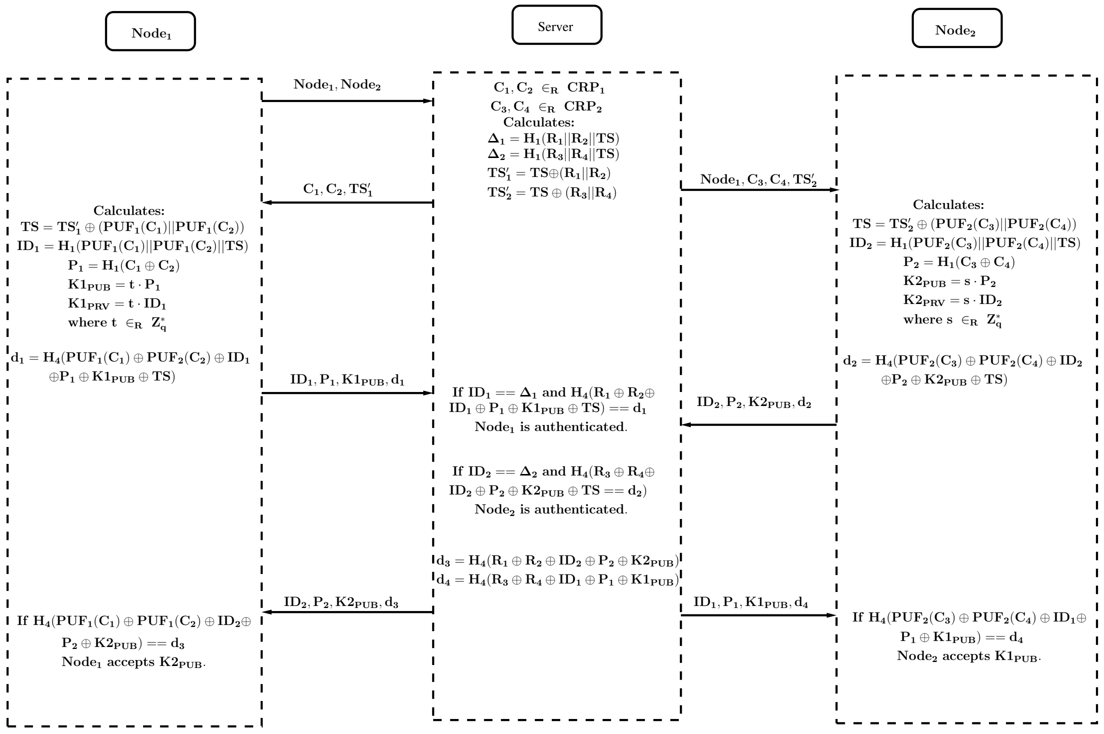
where n is the bit length of the message. So the public mathematical parameters are: $<$$p$, $\mathbb{G}_1$, $\mathbb{G}_2$, ê, n, H_1, H_2, H_3, $H_4$$>$.

3.2. Enrolment Phase

Before deploying the nodes in the communication network, the enrolment phase is executed for each node *in a secure and trusted environment*. The steps are shown in Fig. 3, and are summarized as follows:

- The server sends a random challenge C to the node.
- The node applies the challenge C to its PUF, and generates the output $R = PUF(C)$ and returns it to the server.
- The server stores the response along with the challenge by appending $<C, R>$ to its database.
- This procedure is repeated k times predefined according to the memory capacity of the server.

At the end of the enrolment phase for a given node, the server node supervising the data node has a CRP database with k CRPs for the node. Here we have assumed that the server stores the CRP in a secure database which cannot be accessed by the attacker, and that the server is sufficiently resourceful to implement secure databases. This assumption is based on the fact that there are several existing methodologies [Dubey et al. 2015; Mutti et al. 2015; Kim et al. 2016; Guo and Xu 2015] to control authentication, authorization and grant secure access to databases at the current state-of-the-art.



Node$_1$
Server
Node$_2$
Node$_1$, Node$_2$
$C_1, C_2 \in_R$ CRP$_1$
$C_3, C_4 \in_R$ CRP$_2$
Calculates:
$\Delta_1 = H_1(R_1||R_2||TS)$
$\Delta_2 = H_1(R_3||R_4||TS)$
$TS'_1 = TS \oplus (R_1||R_2)$
$TS'_2 = TS \oplus (R_3||R_4)$
C_1, C_2, TS'_1
Node$_1$, C_3, C_4, TS'_2
Calculates:
$TS = TS'_1 \oplus (PUF_1(C_1)||PUF_1(C_2))$
$ID_1 = H_1(PUF_1(C_1)||PUF_1(C_2)||TS)$
$P_1 = H_1(C_1 \oplus C_2)$
$K1_{PUB} = t \cdot P_1$
$K1_{PRV} = t \cdot ID_1$
where $t \in_R Z_q^*$
$d_1 = H_4(PUF_1(C_1) \oplus PUF_2(C_2) \oplus ID_1 \oplus P_1 \oplus K1_{PUB} \oplus TS)$
Calculates:
$TS = TS'_2 \oplus (PUF_2(C_3)||PUF_2(C_4))$
$ID_2 = H_1(PUF_2(C_3)||PUF_2(C_4)||TS)$
$P_2 = H_1(C_3 \oplus C_4)$
$K2_{PUB} = s \cdot P_2$
$K2_{PRV} = s \cdot ID_2$
where $s \in_R Z_q^*$
$d_2 = H_4(PUF_2(C_3) \oplus PUF_2(C_4) \oplus ID_2 \oplus P_2 \oplus K2_{PUB} \oplus TS)$
$ID_1, P_1, K1_{PUB}, d_1$
$ID_2, P_2, K2_{PUB}, d_2$
If $ID_1 == \Delta_1$ and $H_4(R_1 \oplus R_2 \oplus ID_1 \oplus P_1 \oplus K1_{PUB} \oplus TS) == d_1$
Node$_1$ is authenticated.
If $ID_2 == \Delta_2$ and $H_4(R_3 \oplus R_4 \oplus ID_2 \oplus P_2 \oplus K2_{PUB} \oplus TS == d_2)$
Node$_2$ is authenticated.
$d_3 = H_4(R_1 \oplus R_2 \oplus ID_2 \oplus P_2 \oplus K2_{PUB})$
$d_4 = H_4(R_3 \oplus R_4 \oplus ID_1 \oplus P_1 \oplus K1_{PUB})$
$ID_2, P_2, K2_{PUB}, d_3$
$ID_1, P_1, K1_{PUB}, d_4$
If $H_4(PUF_1(C_1) \oplus PUF_1(C_2) \oplus ID_2 \oplus P_2 \oplus K2_{PUB}) == d_3$
Node$_1$ accepts $K2_{PUB}$.
If $H_4(PUF_2(C_3) \oplus PUF_2(C_4) \oplus ID_1 \oplus P_1 \oplus K1_{PUB}) == d_4$
Node$_2$ accepts $K1_{PUB}$.

Fig. 4. Authentication and key sharing phase.

3.3. Authentication and Key Sharing Phase

The second phase of this protocol performing authentication and key sharing is described below as shown in Fig. 4.

- At first, $Node_1$ initiates a request to inform the server about its willingness of sending a message to $Node_2$, and to supervise the communication.
- The server chooses two challenges C_1 and C_2 randomly from the stored PUF CRP database of $Node_1$, and C_3 and C_4 randomly from the stored PUF CRP database of $Node_2$. The server node also fixes a timestamp (TS) for initializing the protocol, and performs the following computations:

$$\begin{aligned}\Delta_1 &= H_1(R_1||R_2||TS)\\ \Delta_2 &= H_1(R_3||R_4||TS)\\ TS'_1 &= TS \oplus (R_1||R_2)\\ TS'_2 &= TS \oplus (R_3||R_4)\end{aligned} \tag{10}$$

  where R_1, R_2, R_3 and R_4 are the PUF responses corresponding to the challenges C_1, C_2, C_3 and C_4. Note that Δ_1 and Δ_2 are calculated as elements of $\mathbb{G}_1^*$.
- The server node then sends (C_1, C_2, TS'_1) to $Node_1$ and $(Node_1, C_3, C_4, TS'_2)$ to $Node_2$.
- After receiving the message from server, $Node_1$ calculates:
  $TS = TS'_1 \oplus (PUF_1(C_1)||PUF_1(C_2))$
  $ID_1 = H_1(PUF_1(C_1)||PUF_1(C_2)||TS)$
  $P_1 = H_1(C_1 \oplus C_2)$

Thus $Node_1$ applies the challenges C_1, C_2 to its PUF instance, gets the corresponding responses, and calculates the value of $ID_1 \in \mathbb{G}_1^*$, which ideally must be equal to Δ_1.

- Next, $Node_1$ randomly chooses a value t such that $t \in_R Z_q^*$ and computes:
  $K1_{PUB} = t \cdot P_1$
  $K1_{PRV} = t \cdot ID_1$
  $d_1 = H_4(PUF_1(C_1) \oplus PUF_1(C_2) \oplus ID_1 \oplus P_1 \oplus K1_{PUB} \oplus TS)$
- $Node_1$ then sends a message to the server which contains $(ID_1, P_1, K1_{PUB}, d_1)$. Note that $K1_{PUB}$ and $K1_{PRV}$ act as the public and private key for $Node_1$ respectively.
- $Node_2$ also performs operations similar to $Node_1$:
  $TS = TS_2' \oplus (PUF_2(C_3)||PUF_2(C_4))$
  $ID_2 = H_1(PUF_2(C_3)||PUF_2(C_4)||TS)$
  $P_2 = H_1(C_3 \oplus C_4)$
  Thus $Node_2$ applies the challenges C_3, C_4 to its PUF instance, and gets the corresponding responses, and calculates the value of $ID_2 \in \mathbb{G}_1^*$, which ideally must be equal to Δ_2.
- Next, it randomly chooses a value s such that $s \in_R Z_q^*$ and computes:
  $K2_{PRV} = s \cdot ID_2$
  $K2_{PUB} = s \cdot P_2$
  $d_2 = H_4(PUF_2(C_3) \oplus PUF_2(C_4) \oplus ID_2 \oplus P_2 \oplus K2_{PUB} \oplus TS)$. As in case of $Node_1$, $K2_{PUB}$ and $K2_{PRV}$ act as the public and private key for $Node_2$ respectively.
- $Node_2$ sends a message to the server which contains $(ID_2, P_2, K2_{PUB}, d_2)$.
- After receiving messages from both $Node_1$ and $Node_2$, the server node needs to authenticate them individually, and also check whether the public key truly belongs to the correct node claiming it. To ensure this, it first checks if $ID_1 = \Delta_1$, and if the test passes, then data node $Node_1$ is authenticated. Additionally, if $H_4(R_1 \oplus R_2 \oplus ID_1 \oplus P_1 \oplus K1_{PUB} \oplus TS) = d_1$, then the public key is also authentic and it belongs to $Node_1$. Hence the server node accepts it.
- Similarly, if the server finds $ID_2 = \Delta_2$, then the data node $Node_2$ is authenticated. Additionally, if $H_4(R_3 \oplus R_4 \oplus ID_2 \oplus P_2 \oplus K2_{PUB} \oplus TS) = d_2$, then the public key is also authentic and it belongs to $Node_2$. Hence the server node accepts it.
- When the server completes authentication for both the parties, it transmits the public keys to their corresponding receivers. For that, it calculates:
  $d_3 = H_4(R_1 \oplus R_2 \oplus ID_2 \oplus P_2 \oplus K2_{PUB})$ $d_4 = H_4(R_3 \oplus R_4 \oplus ID_1 \oplus P_1 \oplus K1_{PUB})$
- Finally, the server node sends $Node_1$ and $Node_2$ two messages containing $(ID_2, P_2, K2_{PUB}, d_3)$ and $(ID_1, P_1, K1_{PUB}, d_4)$, respectively.
- Once the message from the server node is received by $Node_1$, it calculates: $H_4(PUF_1(C_1) \oplus PUF_1(C_2) \oplus ID_2 \oplus P_2 \oplus K2_{PUB})$. If it equals d_3, then $Node_1$ is ensured that the message is truly generated by the server node, and thus $Node_1$ accepts $Node_2$'s public key.
- Similarly, $Node_2$ calculates: $H_4(PUF_2(C_3) \oplus PUF_2(C_4) \oplus ID_1 \oplus P_1 \oplus K1_{PUB})$. If it equals to d_4, then $Node_2$ is assured that the message is truly generated by the server node, and $Node_1$'s public key is accepted by $Node_2$.

3.4. Secure Communication Phase

The final phase is where the actual secure transmission of message takes place from $Node_1$ to $Node_2$. In this phase, we have used Weil pairing to make the message communication secure against *CPA and CCA attack*. The proof of security is given in [Boneh and Franklin 2003]. In addition, we have modified the protocol so that it becomes secure against *repudiation attack*. In this case, we have utilized the bilinearity property of Weil pairing and collision resistant hash function to ensure that the sender of the

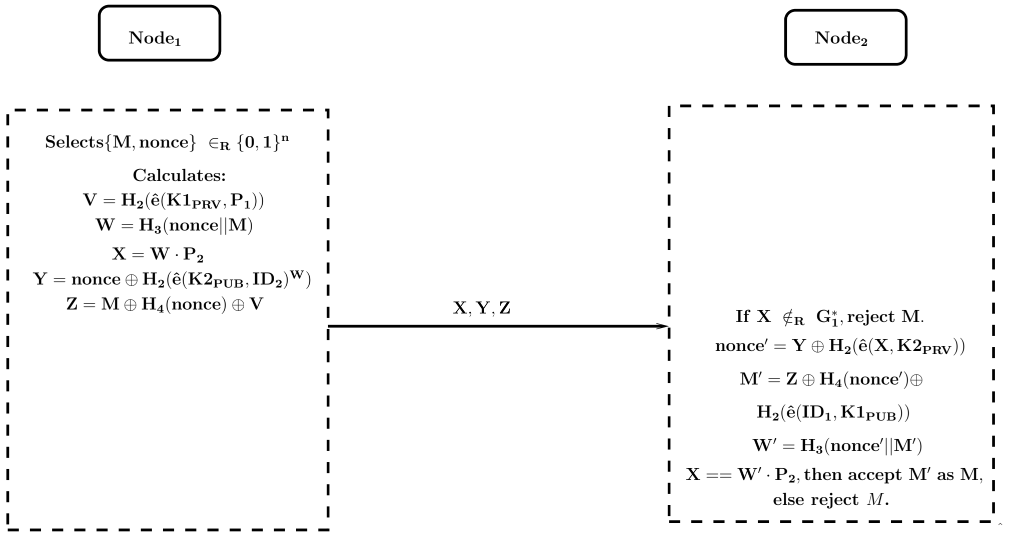

Node$_1$
Node$_2$
Selects{M, nonce} $\in_R \{0,1\}^n$
Calculates:
$V = H_2(\hat{e}(K1_{PRV}, P_1))$
$W = H_3(nonce||M)$
$X = W \cdot P_2$
$Y = nonce \oplus H_2(\hat{e}(K2_{PUB}, ID_2)^W)$
$Z = M \oplus H_4(nonce) \oplus V$
X, Y, Z
If $X \notin_R G_1^*$, reject M.
$nonce' = Y \oplus H_2(\hat{e}(X, K2_{PRV}))$
$M' = Z \oplus H_4(nonce') \oplus$
$H_2(\hat{e}(ID_1, K1_{PUB}))$
$W' = H_3(nonce'||M')$
$X == W' \cdot P_2$, then accept M' as M, else reject M.

Fig. 5. Secure communication phase resistant against CCA and CPA attacks.

message cannot deny it later. In Fig. 5, we have illustrated the steps of the secure communication phase. The secure communication phase consists of the following steps:

- First, $Node_1$ (the sender) executes the following steps:
  - It selects the message M and a nonce. Both of the entities are n-bit long. It then calculates the following: $V = H_2(\hat{e}(K1_{PRV}, P_1))$. Note that since $K1_{PRV} = t \cdot ID_1$, it follows that: $\hat{e}(K1_{PRV}, P_1) = \hat{e}(t \cdot ID_1, P_1) = \hat{e}(ID_1, P_1)^t \in \mathbb{G}_2$.
  - $Node_1$ then calculates $W = H_3(nonce \parallel M)$ and $X = W \cdot P_2$.
  - $Node_1$ further calculates: $Y = nonce \oplus H_2(\hat{e}(K2_{PUB}, ID_2)^W) = nonce \oplus H_2(\hat{e}(s \cdot P_2, ID_2)^W) = nonce \oplus H_2(\hat{e}(P_2, ID_2)^{s \cdot W})$, and,
  - $Z = M \oplus H_4(nonce) \oplus V$
  - Finally, for the plaintext M, $Node_1$ creates the ciphertext as the 3-tuple (X, Y, Z), and sends to $Node_2$.
- Once $Node_2$ receives the ciphertext, it follows the below steps:
  - If $X \notin G_1^*$, $Node_2$ rejects the message, otherwise calculates the following three values:
  - $nonce' = Y \oplus H_2(\hat{e}\,(X, K2_{PRV})$
    $= Y \oplus H_2(\hat{e}\,(W \cdot P_2, s \cdot ID_2)$
    $= Y \oplus H_2(\hat{e}(P_2, ID_2)^{s \cdot W})$
  - $M' = Z \oplus H_4(nonce') \oplus H_2(\hat{e}(ID_1, K1_{PUB}))$
    $= Z \oplus H_4(nonce') \oplus H_2(\hat{e}(ID_1, t \cdot P_1))$
    $= Z \oplus H_4(nonce') \oplus H_2(\hat{e}(ID_1, P_1)^t)$ and,
  - $W' = H_3(nonce' \parallel M')$.
  - If X equals $W' \cdot P_2$, $Node_2$ accepts M' as the message, otherwise it rejects the message.
  - Now, if $Node_1$ does not send the message, then the value of $H_2(\hat{e}(K1_{PUB}, ID_1))$ cannot be equal to V (with only a negligible probability of collision). That, in turn, implies that $X \neq W' \cdot P_2$. Therefore, $Node_2$ never accepts an incorrect message M' in place of M. And if it accepts the message, then $Node_1$ cannot deny that it did not send it.

4. SECURITY ANALYSIS

Attack Models: Next we will shortly describe attack models for the session key exchange protocol. For that, we have considered three different attack models as described below.

- ***Session Key Security Model*:** Under this framework, we have assumed all of the data nodes, servers, routers and network gateway are trusted. The attacker either (i) eavesdrops the communication link without any change or addition to the messages (e.g. packet sniffing attack) or, (ii) has full control over the links and can modify the messages (e.g. packet injection or re-routing attack). In Section 4.4, it has been shown that the protocol is secure against both of these attack variants.
- ***Universally Composite Framework*:** This model ensures that the proposed key exchange protocol provides the same security when used by any other protocol to set up session keys between two parties, even when it runs in parallel with an arbitrary set of other protocols in a distributed communication network. The primary concepts of this framework are described in Section 4.5. We have described an ideal functionality of the asymmetric key exchange protocol in Section 4.6. Here also the attacker has full control over the links. Moreover, she can get more advantages if some of the participants get compromised. We have shown three such scenarios where:
  1. Server and the two communicating parties are honest (ideal case).
  2. The server is corrupt (e.g., the attacker can have access to the CRP database for a limited time)
  3. Either of the two communicating parties or both are corrupt (e.g., in real life implementation, we can picture this scenario as the attacker can control the internal functioning of the node and tries to send some malicious information to disrupt the system)

  Further, we have shown what the implications will be in the proposed scheme for these three level of attacks. We have presented the security analysis for the server-node layer. But it is also applicable for router-server and gateway-router layer also.
- ***Malicious PUF Model*:** In this model, we have assumed that after the enrolment phase, the attacker has either tampered the PUF of a specific node or has replaced it by some malicious or untrusted PUF. Subsequently the attacker eavesdrops or modifies the messages transmitted over the network and tries to guess the private key of that particular node. In this case also, we have shown that our protocol is secure from any such vulnerability.

4.1. Definition of Session-Key Security

The definition of Session-Key Security (SK security) is based on the approach called "security by indistinguishability". To elaborate, this approach evaluates the security of a cryptographic system as follows. Suppose, two games $Game_1$ and $Game_2$ are constructed in which the adversary communicates with the protocol under consideration. If no feasible adversary can distinguish between whether she is interacting with $Game_1$ or $Game_2$, then the protocol is said to be indistinguishable and secure. Further, in order to ensure that the proposed cryptographic scheme is secure against differing capabilities of the attacker, usually two adversarial models are considered:

- **The Unauthenticated-link Adversarial Model (UM)**: Here, a probabilistic polynomial-time (PPT) attacker is considered who has full access/control over the communication links, along with the scheduling of all protocol events such as initiation of protocols and message delivery.

- **The Authenticated-link Adversarial Model (AM)**: In this case, the attacker is restricted to only deliver messages truly generated by the parties without any change or addition to them.

To define UM, first an "experiment" is defined where the attacker Λ chooses to attack a session under "test", and is asked to distinguish between the real value of the session key and a random value. Let κ be the shared session key of the session under test. We consider the result of a coin toss b, where $b \in \{0, 1\}$. If $b = 0$, the value κ is given to the attacker Λ, otherwise a random value r, randomly chosen from the probability distribution of keys generated by the protocol π, is provided. The attacker have the permission to act as a regular UM attacker, and at the end of its run, outputs a bit b'. Under this model,

Definition 4.1. A key-exchange (KE) protocol π is called SK-secure if the following properties hold for any KE-adversary Λ in the UM:

(1) Protocol π satisfies the property that if two uncorrupted parties successfully complete a session then they both output the same key, and,
(2) the probability that Λ guesses correctly the bit i.e., $b' = b$ is more than $\frac{1}{2}$ by only a *negligible quantity*.

We can define a function f as negligible if for every polynomial $p(\cdot)$, there exists an M such that for all integers $n > M$, $f(n) < \frac{1}{p(n)}$ [Katz and Lindell 2007]. In other word, a function that grows slower than any inverse polynomial is termed "negligible".

In our proposed protocol, since we have used the identity-based encryption system, hence the session keys in this case are asymmetric keys. Still, in the subsequent sections we will show that our cryptographic constructions can be extended to be relevant to the above definitions of security, and hence it is SK-secure in AM and UM adversarial model.

4.2. The Uniqueness Property of Physical Unclonable Functions

The cryptographic security of our proposed authentication and key exchange scheme is based on the *Uniqueness Property* of the *Physically Unclonable Functions*. The uniqueness property of the PUF circuit embedded in a chip provides the capability of uniquely identify it from a set of PUF instances of the same type, which have gone through the same manufacturing process. The uniqueness metric is defined as:

$$Uniqueness = \frac{2}{k(k-1)} \sum_{i=1}^{k} \sum_{j=i+1}^{k} \frac{HD(R_i, R_j)}{n} \times 100 \qquad (11)$$

where $HD(R_i, R_j)$ is the *Hamming Distance* between the responses of i^{th} and j^{th} FPGA chip embedding PUF for a particular challenge C and k is the total number of chips under consideration, and n is the total number of response bits of the PUF. The above equation needs some explanation. The metric is calculated considering pairs of PUF instances (P_i, P_j) (with $i \neq j$), calculating the HD between their responses (for the same set of applied challenges), dividing the HD by the number of PUF output bits (i.e. n), summing all the HD values for all the PUF instance pairs, and finally dividing by the total number of (unordered) pairs from the set of k PUF instances (which is $\binom{k}{2}$). The quantity is multiplied by 100 to convert it to a percentage value. The ideal value of the *uniqueness* metric is 50%.

At the current state-of-the-art, it is infeasible to physically clone a given PUF instance, for most PUF types. The only reported successful physical cloning of PUF is [Helfmeier et al. 2013] for the SRAM PUF. In other words, given a PUF instance with a particular CRP characteristics, it is infeasible to fabricate another PUF instance which is sufficiently similar in its CRP characteristics. PUFs can also be mathematically replicated, whereby a mathematical model of a given PUF instance, usually built by machine learning techniques, is capable of predicting the response for an arbitrary challenge with very high probability of success (> 0.95). Certain PUF types (e.g. *Arbiter PUF*) can be mathematically modeled extremely accurately [Rührmair et al. 2010]. **However, by choosing the appropriate PUF type (e.g. *Lightweight Secure PUF* or *XOR PUF*) [Rührmair et al. 2010], we can avoid both physical and mathematical cloning**. We will prove that our proposed protocols are secure as long as the underlying problem of replicating (either physically or mathematically) the challenge-response mapping of a given PUF instance is hard. The basic security assumption is: this challenge-response mapping cannot be constructed accurately by any polynomial time algorithm except with negligible probability, as formalized in the security definitions below:

Definition 4.2. *(Decisional Uniqueness Problem (DUP))* Given a PUF instance PUF_{Adv}, a challenge C and an n-bit string $z \in \{0,1\}^n$, the DUP aims to decide whether $z = PUF_N(C)$ for a PUF instance PUF_N, or a random n-bit string.

Definition 4.3. *(2-Decisional Uniqueness Problem (2-DUP))* Given a PUF instance PUF_{Adv}, two challenges C_1, C_2, and two n-bit strings $z_1, z_2 \in \{0,1\}^n$, the problem aims to find out whether $z_1 = PUF_N(C_1)$ and $z_2 = PUF_N(C_2)$ for another PUF instance PUF_N, or two random n-bit strings.

Next we will define the assumption for DUP and 2-DUP which will provide the computational indistinguishability for some probabilistic, polynomial time algorithm. The computational indistinguishability refers to the *probability ensembles* which are infinite sequence of probability distributions. The Decisional Uniqueness Problem Assumption can be formalized by stating that the ensemble of tuples of type (C, PUF_{Adv}, z) is computationally indistinguishable from the ensemble of tuples of type $(C, PUF_{Adv}, PUF_N(C))$. Similarly, 2-Decisional Uniqueness Problem Assumption can be formalized by stating that the ensemble of tuples of type $(C_1, C_2, PUF_{Adv}, z_1, z_2)$ is computationally indistinguishable from the ensemble of tuples of type $(C_1, C_2, PUF_{Adv}, PUF_N(C_1), PUF_N(C_2))$.

Definition 4.4. *(Decisional Uniqueness Problem Assumption)* The problem of fabricating a PUF instance PUF_N using another instance PUF_{Adv} is hard, and for all probabilistic, polynomial time algorithm $\mathcal{A}$, there exists a *negligible function* $negl(\cdot)$ such that:

$$| \; Pr[\mathcal{A}(C, PUF_{Adv}, z) = 1] - Pr[\mathcal{A}(C, PUF_{Adv}, PUF_N(C)) = 1] \; | \leqslant negl(n) \tag{12}$$

where n is the number of response bits of the PUF instance. This implies that given an arbitrary challenge C and an arbitrary PUF instance PUF_{Adv}, the adversary $\mathcal{A}$ behaves almost identically, for a random element $z \in \{0,1\}^n$, and the actual n-bit response $PUF_N(C)$.

Definition 4.5. *(2-Decisional Uniqueness Problem Assumption)* The problem of fabricating a PUF instance PUF_N using another instance PUF_{Adv} is hard, and for all probabilistic, polynomial time algorithm $\mathcal{B}$, there exists a *negligible function* $negl(\cdot)$ such that:

$$| \; Pr[\mathcal{B}(C_1, C_2, PUF_{Adv}, z_1, z_2) = 1] - \\ Pr[\mathcal{B}(C_1, C_2, PUF_{Adv}, PUF_N(C_1), PUF_N(C_2)) = 1] \; | \leqslant negl(n)$$

which implies that given two challenges C_1, C_2 and the PUF instance PUF_{Adv}, the adversary $\mathcal{B}$ behaves almost identically, for a pair of two random elements $z_1, z_2 \in \{0,1\}^n$, and the actual n-bit responses $PUF_N(C_1)$ and $PUF_N(C_2)$.

Now we prove the following claim:
Claim: The 2-DUP problem is at least as hard as DUP.

PROOF. Suppose, there exists a PPT adversary $\mathcal{B}$ such that $\mathcal{B}$ has non-negligible advantage ε in solving 2-DUP, i.e., $\mathcal{B}$ can break 2-DUP with a non-negligible probability ε greater than $\frac{1}{2}$. Then, we will construct a PPT adversary $\mathcal{A}$ which also has advantage ε in solving DUP.

$\mathcal{A}$ takes as input a random DUP input tuple (C,PUF_{Adv},z). Then, $\mathcal{A}$ constructs an 2-DUP input tuple (C, C, PUF_{Adv}, z, z) and gives it to $\mathcal{B}$. If $z = PUF_n(C)$, note that this is a valid input tuple to $\mathcal{B}$. The behaviour of $\mathcal{A}$ is such that if $\mathcal{B}$ outputs 1, then $\mathcal{A}$ outputs 1; otherwise $\mathcal{A}$ outputs 0. Since $\mathcal{A}$ perfectly simulates $\mathcal{B}$, hence, $\mathcal{A}$ has the same advantage ε in breaking DUP. Therefore, 2-DUP is at least as hard as solving DUP. □

The implication of the above claim is that if we assume DUP to be a standard hard problem, 2-DUP is also another difficult problem to solve. This result is crucial in proving the security of the proposed protocol.

4.3. Correctness Proof of the Proposed Scheme

We consider a setting with two parties, a data node N and the corresponding server node currently in charge of the data node N. We denote the protocol by π. Recall that the data node N contains a PUF instance PUF_N, and the server node contains a database consisting of a subset of the challenge-response pairs (CRPs) of PUF_N. The protocol also uses the timestamp of the session initiation in the computation to make the approach more granular – this compels the node N to authenticate itself to the server every time it wants to create a session with other nodes. Within a single communication session, these two parties can communicate with each other multiple times.

Let TS denote the timestamp used by the server at the time of the protocol run for each node. Moreover, let $output_{N,\pi}(C_1, C_2, (PUF_N(C_1)||PUF_N(C_2)) \oplus TS)$ and $output_{S,\pi}(C_1, C_2, (R_1||R_2) \oplus TS)$ denote the respective outputs of the data node N, and the server node (upon receiving the challenges C_1, C_2, and the time stamp TS). We assume that this output takes the form of an element of G_1^* that is supposed to be considered as the identity of node N, and should be shared by the data node N and the server node. Hence,

$$output_{N,\pi}(C_1, C_2, (PUF_N(C_1)||PUF_N(C_2)) \oplus TS) = H_1(PUF_N(C_1)||PUF_N(C_2)||TS) \tag{13}$$

and

$$output_{S,\pi}(C_1, C_2, (R_1||R_2) \oplus TS) = H_1(R_1||R_2||TS) \tag{14}$$

Next, we present the definition of the correctness requirement. It states that, except with negligible probability, the data node N and the server node will generate the same identity, and only node N will be authenticated to the server.

Definition 4.6. A protocol π for authentication and key exchange is called *correct* if there exists a negligible function $negl(\cdot)$, such that for every possible value of n:

$$\Pr[output_{N,\pi}(C_1, C_2, (PUF_N(C_1)||PUF_N(C_2)) \oplus TS) \neq output_{S,\pi}(C_1, C_2, (R_1||R_2) \oplus TS)] \leqslant negl(n)$$

It can be trivially observed that:

$$H_1(PUF_N(C_1)||PUF_N(C_2)||TS) = H_1(R_1||R_2||TS) \tag{15}$$

This means that the data node N and the server node will output the same value, thereby proving the correctness of the scheme.

4.4. Security Proof of the Proposed Scheme

Now we define security in the context of the proposed authentication and key exchange protocol in general. Intuitively, an authentication and key exchange protocol is secure if the identity output by the data node N and the server node are identical, and no adversary can correctly guess the identity for two challenges C_1 and C_2 and a timestamp TS chosen randomly. This has been formulated by giving an adversary the two particular challenges C_1, C_2 and TS' of a protocol execution, and observing if she can distinguish between the ID output by data node N, and the server node and a completely random element of G_1^*. We would show that breaking the proposed protocol is at least as difficult as solving the 2-Decisional Uniqueness Problem, i.e., a successful attack on the proposed protocol implies a feasible solution to the 2-Uniqueness Problem. In order to demonstrate this, an experiment has been presented next.

Let *Adv* be a probabilistic, polynomial time adversary, and the number of PUF response bits be n. Then, consider the following experiment:

The Eavesdropping Authentication and Key Exchange Experiment $\mathbf{Auth_{adv,\pi}(n, \zeta, PUF_{Adv}, ID_0, ID_1)}$:

(1) The adversary *Adv* is provided:
   (a) ζ=$< C_1, C_2, TS' >$ where $TS' = ((PUF_N(C_1)||PUF_N(C_2)) \oplus TS)$.
   (b) A PUF instance PUF_{Adv}.
   (c) two identities ID_0 and ID_1, which are calculated as: a random bit $b \in \{0, 1\}$ is chosen and the followings have been calculated.

$$ID_b = H_1(PUF_N(C_1)||PUF_N(C_2)||TS)$$
$$ID_{1-b} = h \in_R G_1^*$$

(2) The adversary *Adv* will output a value b'.

The adversary *Adv* succeeds in the experiment if she can distinguish between the "correct" ID and the random one. Next we will prove the following theorem.

THEOREM 4.7. *The authentication and key exchange protocol π is secure in the presence of eavesdropping adversaries if the 2-Decisional Uniqueness Problem Assumption holds.*

PROOF. To prove this, we will show that the protocol π is secure if the adversary succeeds in the experiment $\mathbf{Auth_{adv,\pi}}$ with probability that is at most negligibly greater than $\frac{1}{2}$, i.e., for every probabilistic polynomial time adversary *Adv*, there exists a negligible function $negl(\cdot)$ such that:

$$Pr[Auth_{adv,\pi} = 1] \leqslant \frac{1}{2} + negl(n)$$

Let us assume that the adversary *Adv* has some non-negligible advantage ε in breaking the protocol π. Then we can construct an algorithm $\mathcal{B}$ which will have the same advantage ε in breaking the 2-Uniqueness problem. Now the algorithm $\mathcal{B}$ takes as input a random 2-Uniqueness Problem tuple $(C_1, C_2, PUF_{Adv}, z_1, z_2)$ (where $z_1 = PUF_N(C_1)$ and $z_2 = PUF_N(C_2)$ or two random string belongs to $\{0, 1\}^*$) and proceeds as follows:

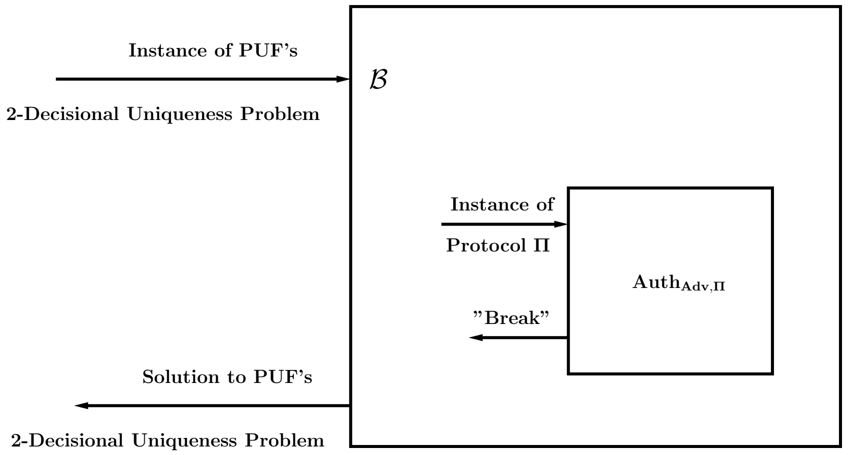

Instance of PUF's
2-Decisional Uniqueness Problem
$\mathcal{B}$
Instance of
Protocol Π
$\mathbf{Auth}_{\mathbf{Adv},\Pi}$
"Break"
Solution to PUF's
2-Decisional Uniqueness Problem

Fig. 6. The view of $\mathbf{Auth}_{\mathbf{adv},\pi}$ when it is run as a sub-routine of $\mathcal{B}$ (referred to [Katz and Lindell 2007]).

(1) **SetUp**:Provide *Adv* with PUF_{Adv}.
(2) **Input tuple**:
  (a) First it randomly chooses TS.
  (b) It calculates $TS' = (z_1||z_2) \oplus TS$.
  (c) Then sets $\zeta = <C_1, C_2, TS'>$, which is perfectly random to the adversary *Adv*.
  (d) Next, it randomly chooses $b \in \{0, 1\}$.
  (e) It then calculates $ID_b = H_1(z_1||z_2||TS)$ and $ID_{1-b} = h \in_R G_1^*$
  (f) The algorithm $\mathcal{B}$ finally provides *Adv* the input tuple $< \zeta, ID_0, ID_1 >$. If $z_1 = PUF_N(C_1)$ and $z_2 = PUF_N(C_2)$, then ID_b will be equal to $H_1(PUF_N(C_1)||PUF_N(C_2)||TS)$ and it will a valid input tuple. Otherwise, ID_0, ID_1 both will be some random element of G_1^*.
(3) **Guess**: The adversary *Adv* returns b', a guess of b. If $b = b'$, then the algorithm $\mathcal{B}$ returns 1, implying that z_1, z_2 are the correct responses of C_1, C_2. Otherwise, it returns 0.

Hence, it is proved that the adversary *Adv* has the same advantage ε as the adversary $\mathcal{B}$. But, due to the hardness 2-Uniqueness Problem, ε should be negligible. Hence,

$$Pr[Auth_{adv,\pi} = 1] \leqslant \frac{1}{2} + negl(n)$$

□

Once the authentication is done successfully, the data node N selects a random value $t \in_R Z_q^*$. Then, it locally calculates its {public,private} key pair $K1_{PUB} = t \cdot P_1$ and $K1_{PRV} = t \cdot ID_1$, where $P_1 = H_1(C_1 \oplus C_2)$. It keeps $K1_{PRV}$ secret and sends $K1_{PUB}$ to the server over the authenticated link. Now assuming the complexity of the *Computational Discrete Log Problem*, the probability that the adversary *Adv* can retrieve the value of t from $K1_{PUB}$, knowing the value of ID_1 is negligible. Hence the adversary *Adv* fails to calculate the correct value of private key $K1_{PRV}$.

If we consider the AM adversarial model, the adversary *Adv* is restricted to only deliver messages truly generated by the parties without any change or addition to them; hence she fails to calculate the private key of node N. On the other hand, in the UM adversarial model, any change in the message sent over the channel will end up with difference in the hashed value of the message at the data node and sever node ends. From the result obtained in the previous theorem, we conclude the following:

THEOREM 4.8. *Based on the complexity assumption of the Computational Discrete Log Problem and that the hash function is collision resistant, the authentication and key-exchange protocol* π *is* SK-secure *in AM as well as in UM model.*

Next, we will prove the compatibility of the scheme with the universally composable framework.

4.5. Security under Universally Composable Framework

First we turn to familiarize with the concept of *Universally Composable* (UC) framework. The basic objective of this framework is to guarantee that any key exchange protocol provides the same security for any other protocol which wants to set up session keys between two parties, even when it runs in parallel with an arbitrary set of other protocols in a distributed communication network. We will prove that the method of key exchange as proposed in this work is also compatible with similar composability properties. The definition of UC framework follows the approach referred as "security by emulation of an ideal process". The primary concept of this principle is as given below [Canetti 2001]:

(1) The model of protocol execution consists of the communicating parties running the protocol and the adversary. They are further considered as interacting computing elements and modelled as *Interactive Turing Machines* (ITMs).
(2) We formulate an "ideal process" $\mathcal{F}$ that picks up the task f of the desired functionality.
(3) In the ideal process $\mathcal{F}$ all communicating parties provides inputs to an "idealized trusted party" which locally performs the task, and sends each party its desired output. In this regard, it is the *formal specification* of the security requirements of the task.
(4) Additionally, a new algorithmic object, called the "environment machine" $\mathcal{E}$, is added in this computational model, which is considered to consist of everything external to the current protocol execution, such as other protocol executions and their adversaries, human users, etc.
(5) The adversary Λ can directly interact with $\mathcal{E}$ throughout the execution of the protocol. They can exchange information after each message or output generated by a party running the protocol. The environment $\mathcal{E}$ also has the permission to apply inputs to the communication parties, and collect outputs from them. But the environment $\mathcal{E}$ is constrained to collect outputs of the main program running in each party, and not the output from the subroutines called from that main program. For example, in our protocol, if the environment $\mathcal{E}$ randomly selects two challenges and a timestamp and applies to any data node $Node_i$, it will receive $(ID_i, P_i, Ki_{PUB}, d_i)$ as output (refer to Fig. 4). But it cannot directly query the PUF instance of $Node_i$ for a particular challenge and collect the response.
(6) If any protocol π securely realizes any task f with respect to this kind of "interactive environment", then it is said that "π UC-realizes f".
(7) More illustratively, we can say that a protocol π *securely realizes the task* if the running protocol π *emulates* the ideal process $\mathcal{F}$ for the task f, i.e., if there exists an adversary Λ which attacks protocol π, there also exists a "simulator" $\mathcal{S}$ that achieves similar adversarial effect by interacting with the ideal process $\mathcal{F}$. In addition, no environment $\mathcal{E}$ can tell with non-negligible probability of success whether it is interacting with Λ and π, or with $\mathcal{S}$ and $\mathcal{F}$ for f.

In the paper [Brzuska et al. 2011], the authors presented the ideal functionality concept for symmetric key exchange protocol. In the following subsections, we will extend these concepts for our asymmetric key exchange phase, and will also prove that our scheme *securely realizes* the ideal functionality.

4.6. UC Security of the Proposed Key Exchange Phase

The main concept of asymmetric key exchange ideal functionality $\mathcal{F}_{AKE}$ is the following:

(1) If both the communicating parties are honest, the functionality provides them with two random IDs, which is written directly to the party's input tape. The adversary cannot have access to the tape, hence the values are invisible to her.
(2) If one of the communicating parties is corrupt, then the adversary can easily determine the ID of the corrupt party.
(3) $\mathcal{F}_{AKE}$ is parameterized by an integer N (the total number of permissible sessions), where a server node runs with exactly n parties, $Node_1$, $Node_2$,...,$Node_n$ and the simulator $\mathcal{S}$.

The working principle of $\mathcal{F}_{AKE}$ has been shown the Fig. 7. Next we prove the security of $\mathcal{F}_{AKE}$.

THEOREM 4.9. *Protocol π securely realizes functionality $\mathcal{F}_{AKE}$.*

PROOF.
Here we assume that (a) the adversary possesses two PUF instances; (b) queries to the PUFs are genuinely handed on to the simulator $\mathcal{S}$'s PUFs, and (c) the PUFs' answers are forwarded unmodified to the querying party throughout all the simulations. We consider different usage and security scenarios in turn.
Case-1: Server and the two communicating parties are honest:

- **Setup:** Whenever the functionality $\mathcal{F}_{AKE}$ receives message $(\mathbf{establish-session_{AKE}}, \mathbf{sid}, \mathbf{Node_i}, \mathbf{server}, \mathbf{Node_j})$ for the first time, the simulator $\mathcal{S}$ queries the two PUF instances PUF_i and PUF_j for k random challenges C_1, C_2,..., C_k, and obtains responses R_{i1}, R_{i2},..., R_{ik} and R_{j1}, R_{j2},..., R_{jk}. Then, it creates two lists $\mathcal{L}_i$ and $\mathcal{L}_j$ of k challenge-response pairs.
- It then hands over PUF_i and PUF_j to $Node_i$ and $Node_j$.
- On receiving a message $(\mathbf{establish-session_{AKE}}, \mathbf{sid}, \mathbf{Node_i}, \mathbf{server}, \mathbf{Node_j})$, $\mathcal{F}_{AKE}$ increments p by one and the simulator $\mathcal{S}$ sends $(deliver_{AKE}, sid, server)$ to $\mathcal{F}_{AKE}$.
- $\mathcal{F}_{AKE}$ then sends $(deliver_{AKE}, sid, ID_i, ID_j, server)$ to the server.
- Now the simulator $\mathcal{S}$ is activated again and it simulates that the server sends (C_1, C_2, TS_1') and (C_3, C_4, TS_2') to $Node_i$ and $Node_j$.
- When the adversary Λ instructs to send the latter message to $Node_i$ and $Node_j$, the simulator $\mathcal{S}$ sends $(deliver_{AKE}, sid, ID_i, Node_i)$ and $(deliver_{AKE}, sid, ID_j, Node_j)$ to $\mathcal{F}_{AKE}$.
- The probability that the value of $H_1(PUF_i(C_1)||PUF_i(C_2)||TS_i)$ is equal to ID_i and the value of $H_1(PUF_j(C_3)||PUF_j(C_4)||TS_j)$ is equal to ID_j, is *negligible* (as proved in Section 4.4) , where $TS_i = (PUF_i(C_1)||PUF_i(C_2)||TS_1')$ and $TS_j = (PUF_j(C_3)||PUF_j(C_4)||TS_2')$.

Case-2: Server is corrupt:

- The simulator $\mathcal{S}$ lets the server to instantiate two PUFs PUF_i and PUF_j, and hands them over to $Node_i$ and $Node_j$.

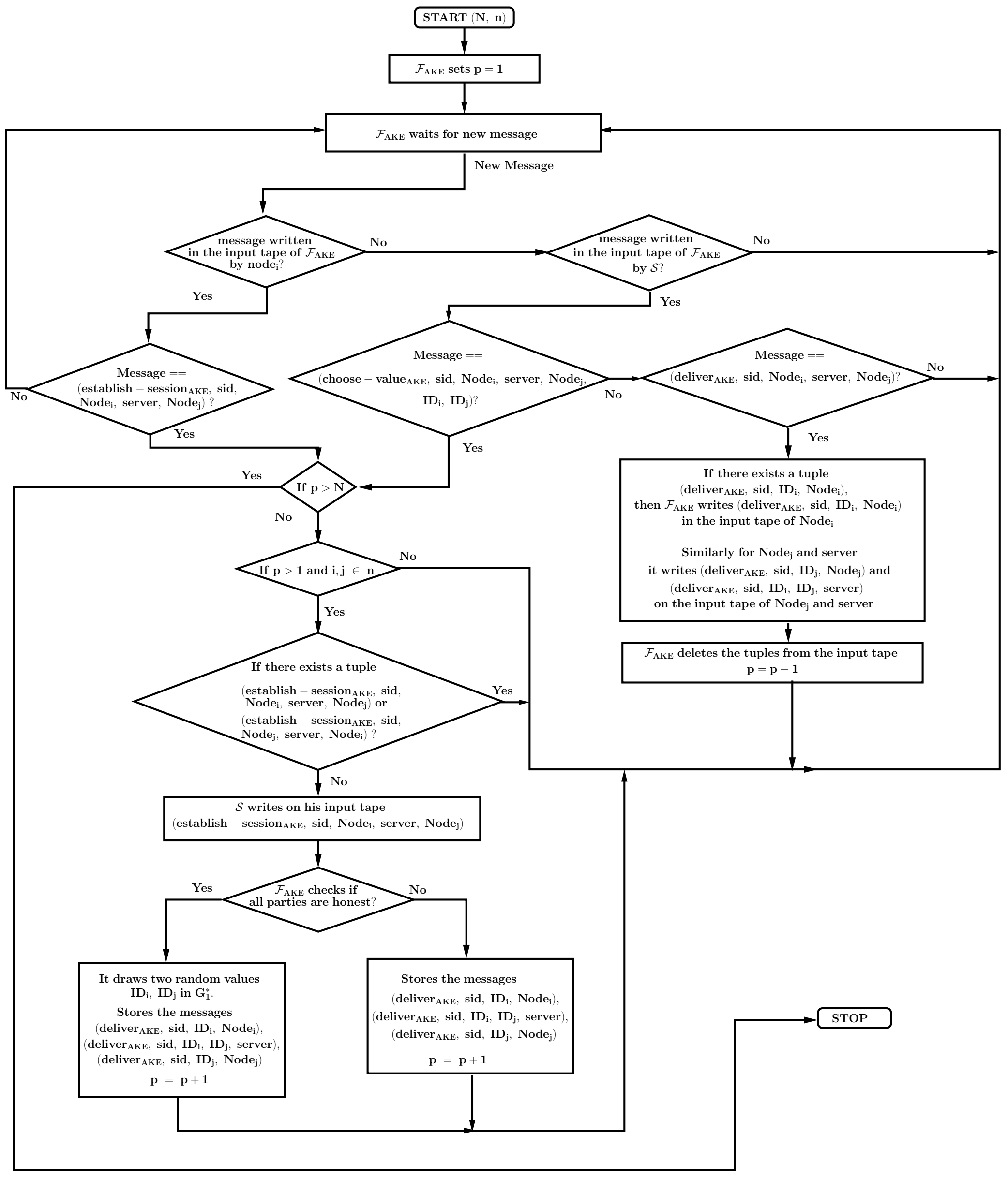

START (N, n)
$\mathcal{F}_{AKE}$ sets p = 1
$\mathcal{F}_{AKE}$ waits for new message
New Message
message written in the input tape of $\mathcal{F}_{AKE}$ by node$_i$?
No
message written in the input tape of $\mathcal{F}_{AKE}$ by $\mathcal{S}$?
No
Yes
Yes
Message == (establish − session$_{AKE}$, sid, Node$_i$, server, Node$_j$) ?
Message == (choose − value$_{AKE}$, sid, Node$_i$, server, Node$_j$, ID$_i$, ID$_j$)?
Message == (deliver$_{AKE}$, sid, Node$_i$, server, Node$_j$)?
No
No
No
Yes
Yes
Yes
If p > N
Yes
No
If there exists a tuple (deliver$_{AKE}$, sid, ID$_i$, Node$_i$), then $\mathcal{F}_{AKE}$ writes (deliver$_{AKE}$, sid, ID$_i$, Node$_i$) in the input tape of Node$_i$
Similarly for Node$_j$ and server it writes (deliver$_{AKE}$, sid, ID$_j$, Node$_j$) and (deliver$_{AKE}$, sid, ID$_i$, ID$_j$, server) on the input tape of Node$_j$ and server
If p > 1 and i, j ∈ n
No
Yes
$\mathcal{F}_{AKE}$ deletes the tuples from the input tape p = p − 1
If there exists a tuple (establish − session$_{AKE}$, sid, Node$_i$, server, Node$_j$) or (establish − session$_{AKE}$, sid, Node$_j$, server, Node$_i$) ?
Yes
No
$\mathcal{S}$ writes on his input tape (establish − session$_{AKE}$, sid, Node$_i$, server, Node$_j$)
Yes
$\mathcal{F}_{AKE}$ checks if all parties are honest?
No
It draws two random values ID$_i$, ID$_j$ in G$_1^*$. Stores the messages (deliver$_{AKE}$, sid, ID$_i$, Node$_i$), (deliver$_{AKE}$, sid, ID$_i$, ID$_j$, server), (deliver$_{AKE}$, sid, ID$_j$, Node$_j$) p = p + 1
Stores the messages (deliver$_{AKE}$, sid, ID$_i$, Node$_i$), (deliver$_{AKE}$, sid, ID$_i$, ID$_j$, server), (deliver$_{AKE}$, sid, ID$_j$, Node$_j$) p = p + 1
STOP

Fig. 7. The asymmetric key exchange ideal functionality.

- When the adversary Λ instructs to deliver messages (C_1, C_2, TS'_1) and (C_3, C_4, TS'_2), then the simulator $\mathcal{S}$ can easily evaluate $ID_i = H_1(PUF_i(C_1)||PUF_i(C_2)||TS_i)$ and $ID_j = H_1(PUF_j(C_3)||PUF_j(C_4)||TS_j)$, as the server is corrupt.
- It next sends $(choose - value_{AKE},\ sid,\ Node_i,\ server,\ Node_j,\ ID_i,\ ID_j)$ to $\mathcal{F}_{AKE}$ as it has already calculated the value of ID_i and ID_j and $\mathcal{F}$ increments the value of p by one.
- Finally, $\mathcal{S}$ sends the messages $(deliver_{AKE},\ sid,\ ID_i,\ Node_i)$ and $(deliver_{AKE},\ sid,\ ID_j,\ Node_j)$ to $\mathcal{F}_{AKE}$.

- Hence in this case the ID provided by $\mathcal{F}$ and the ID calculated from the challenges given by the server is same.
- But $Node_i$ later chooses a random value $t \in Z_p^*$ after getting the ID_i, and calculates the public and private keys using them. Hence, the simulator $\mathcal{S}$ as well as the adversary Λ cannot guess the asymmetric key pairs for $Node_i$. This is due to fact that the security of elliptic curve cryptography rests on the assumption that the elliptic curve discrete logarithm problem (ECDLP) is hard. Let E is an elliptic curve over a finite field K and there are points $P, Q \in E(K)$ such that $Q \in < P >$, where P is a primitive point and $< P >$ denotes the set of points generated by adding P k times, for some value of k, i.e., $Q = [k]P$ $(P + P + P ... + P$ k times). The ECDLP problem is to find the value of k given P and Q and it is considered to be a hard problem. Now as $Node_i$ randomly selects the value of t and $P_1, K1_{PUB}$ are the points on the elliptic curve, it is assumed to be hard to predict the value of t by the simulator $\mathcal{S}$ and the adversary Λ. The same is true for $Node_j$.
  So, we can say that even if the server gets corrupted for a limited time, *the keys of the legitimate users are not compromised* which in turn ensures that the *data communicated between two nodes cannot be retrieved by the corrupted server*.

Case-3: Either of the two communicating parties or both are corrupt:

This case covers the situation if a party willingly hands over its PUF to the adversary Λ. So in this case, we will show that the adversary Λ can easily retrieve the value of private key for that particular party. Note that the protocol ensures that it can easily track the party's action, and a malicious party can never deny that a communication has been taken place by it. Hence, it cannot forge the identification of its actions.

- The set up phase is same as given in Case-1.
- On receiving a message $(\mathbf{establish - session_{AKE}},\ \mathbf{sid},\ \mathbf{Node_i},\ \mathbf{server},\ \mathbf{Node_j})$, the simulator $\mathcal{S}$ increments p by one and sends $(choose - value_{AKE},\ sid,\ Node_i,\ server,\ Node_j,\ ID_i,\ ID_j)$ to $\mathcal{F}_{AKE}$.
- It is activated again and sends $(deliver_{AKE},\ sid,\ server)$ to $\mathcal{F}_{AKE}$.
- The server writes the ID_i and ID_j on its local tape and $\mathcal{S}$ is activated again.
- It simulates the server sending (C_1, C_2, TS'_1) and (C_3, C_4, TS'_2) to $Node_i$ and $Node_j$.
- When the adversary Λ instructs to deliver the latter message to $Node_i$ and $Node_j$, $\mathcal{S}$ sends $(deliver_{AKE},\ sid,\ ID_i,\ Node_i)$ and $(deliver_{AKE},\ sid,\ ID_j,\ Node_j)$ to $\mathcal{F}_{AKE}$.
- If $Node_i$ or $Node_j$ or both are corrupted, then Λ can easily find out the random value chosen from Z_p^* for calculating the private and public keys, and hence the value of the private keys are compromised.
- But as shown in Section 3.4, the communication phase ensures non-repudiation as each message provides the proof of integrity and origin of the data.

Hence, the scheme securely realizes the ideal functionality $\mathcal{F}_{AKE}$. □

4.7. Security Evaluation with Respect to Malicious PUF

We will now describe an analysis of our protocol in the new attack model using malicious PUFs proposed in [Ruhrmair and van Dijk 2013]. With this respect, the following assumptions have been made in [Ostrovsky et al. 2013]:

(1) The adversary is allowed to create untrusted or malicious PUFs.
(2) An adversary can tamper a given PUF instance, but such tampering might result in hampering its security as well as irreversible changes in its physical intrinsic properties.

Table I. Comparison of existing and proposed authentication protocols

Protocols	**Advantages and disadvantages of the proposed scheme**
Controlled PUF Protocol [Gassend et al. 2002]	*Issues:* To protect against modelling attack, this protocol requires an huge number of CRPs for the PUF. Therefore, storage requirement is very high. *Advantage:* The number of CRPs used here are moderate, as the response corresponding to a particular challenge is never sent over the network directly. So, the challenges are re-usable.
Noisy PUF Protocol [Öztürk et al. 2008]	*Issues:* Here also, to prevent PUF modelling attack, an internal secret is XOR-ed with the challenge. But to derive a new secret, the old one needs to be remembered *for each instance*. Hence, the synchronization effort is an overhead. Moreover, it may enable denial-of-service (DoS) attack. *Advantage:* To avoid the DoS attack, we have introduced a random parameter, namely the *TimeStamp* (TS), through which the server can keep track of the sessions running between two communicating parties. It implies that no other request from two nodes already being served by a particular server node, will be served by the same server node, until the current session has expired.
Reconfigurable PUF Protocol [Katzenbeisser et al. 2011]	*Issues:* The protocol also implements a pseudorandom number generator (PRNG) which leads to a DoS attack. *Advantage:* As mentioned earlier, we use the timestamp to prevent the protocol against this kind of attack.
Reverse FE Protocol [Herrewege et al. 2012]	*Issues:* The protocol implements a PRNG to expand the PUF response space which leads to token/server impersonation. *Advantage:* As given in Section 4, the proposed scheme is secure against impersonation.
Slender PUF Protocol [Majzoobi et al. 2012]	*Issues:* The protocol employs a PRNG to expand the PUF response space which leads to token impersonation. *Advantage:* The proposed scheme is secure against impersonation as mentioned before.
Converse PUF Protocol [Koçabas et al. 2012]	*Issues:* First, the attacker model is too restricted, as the authors have considered the security only against eavesdropping attacks. Secondly, the server has to perform an exhaustive search over the CRP database of the token for each authentication initiation, which leads to scalability issues. *Advantage:* The proposed scheme has been proved to be secure against passive as well as active attacks also. He have considered the security in AM and UM adversarial model, and shown its compatibility in the UC framework as well. Scalability is not an issue in this case, as the protocol initiation is done by the server itself.

(3) Otherwise, the adversary might deploy some malicious programmable hardware token in the network which may behave like a PUF at its interface, but is actually programmed with some malicious code.
(4) Once the malicious PUF is deployed in the network, it cannot interact with its creator.

One of the major advantages of our protocol is that it avoids exchange of the PUF instance between the sender and receiver. This means that the protocol can be executed without both communicating parties having physical access to the PUF; it is sufficient if one of the parties access to only the PUF CRP database. This is in contrast to the protocol proposed in [Canetti 2001], two honest parties can have the same key only if they have the same honest PUF. Now, the adversary can either query the honest PUF during the handover phase (but in that the success probability is not high for guessing the key in a random session), or it can replace the honest PUF with a fake PUF, thus falling victim to the attacks proposed in [Ostrovsky et al. 2013]. But in our proposed protocol, the adversary does not have this provision. She can sent a fake PUF, but as that PUF is not enrolled in the server, it cannot act as an honest (enrolled) PUF. Due to the unpredictability and uniqueness property of PUFs, and the fact that the timestamp is a randomly generated value, any node is able to answer to such a query only if it possesses that particular enrolled PUF. Hence, we can say that our proposed construct is also secure against the malicious PUF model.

4.8. Comparison with Existing Protocols

Table I provides a comparative study of several previously proposed PUF based authentication protocols with our scheme. Here, we have presented the security issues raised in [Delvaux et al. 2014] for each previously proposed authentication protocol, and pointed out the features of our proposed architecture eliminates these shortcomings.

5. EXPERIMENTAL SET UP AND RESULTS

In this section, we have described the set up of the test bed for realising the protocol and provided the hardware overhead and execution time of the main components of the scheme. To realise this, the COTS that we procured are Intel Edison and Xilinx Artix-7 FPGA. Intel Edison is a deeply embedded IoT computing module with Bluetooth, Ethernet and WiFi connectivity. We can connect different kinds of sensors (e.g. Sound Sensor, Temperature Sensor, Touch Sensor, Light Sensor etc.) and actuators (e.g. buzzer, LED, LCD RGB Back light, Mini Servo) with this board and it has inbuilt libraries to support those I/O and sensor interactions.

Our main target is to use PUF as device authentication. So the PUF instance for each Edison board was implemented in the Nexys-4 board containing a Xilinx Artix-7 FPGA which will be connected to the particular Edison board and they both together forms the *IoT device*. In this setup, we have used PC as the server and the CRP database of each of these *IoT device* will be stored there in the Enrolment phase. At the time of authentication, the PC sends the challenges to the Edison board through WiFi. The Edison board applies the challenge to the FPGA, collects the responses and send them back to the PC. The communication between FPGA and Edison board is established through UART (Universal Asynchronous Receive/Transmit) protocol.

The PUF design and implementation was performed using Xilinx ISE (v 14.2) design environment. We have used the basic concepts of lightweight secure PUFs proposed by [Majzoobi et al. 2008]. We have used 6-stage parallel PUFs and instead of using output network, we have XOR-ed the responses from these 6 stages. For each row of the parallel PUFs, we have implemented the classical Arbiter PUFs with 64 switches, thereby resulting in 64 bit challenges. The reliability and uniformity of the output bit is 96.11% and 49.5% for 64 bit LSPUF where the uniformity measurement has been taken using 100000 CRPs. To generate 48 bit response string and to improve the statistical quality of the responses, we used a Linear Feedback Shift Register (LFSR) at the input. Next, for a particular challenge C_i, the LFSR is loaded with the challenge as a seed and after 10 clock cycles, the output state of the LFSR was input to the LSPUF. The output state of the LSPUF was saved in a shift register. This process has been executed for 48 times to get a 48 bit response R_i from the shift register for the challenge C_i. By following these steps, the CRP database of each IoT device was created and saved in the secure server. The hardware overhead for deploying the PUF in the FPGA has been shown in Table II.

Next, we have used the MIRACL Crypto SDK [MIR] which provides a C software library for elliptic curve cryptography. We have implemented necessary libraries which are required to define the elliptic curve and to calculate Tate pairing in the Edison board. Along with this, the elliptic curve definition is also required for the calculating

Table II. Hardware overhead of main components of the PUF

Components	No. of Slices	No. of Registers	No. of LUTs
6-Stage 64 Bit LSPUF	776	12	986
64 Bit LFSR	16	48	6
48 Bit Shift Register	15	48	3

Table III. Execution time for different software modules in Edison Board

Components	Execution Time (in sec)	Clock Cycles
Tate Pairing	0.02019	100950
H_1	0.071886	359430
H_2	0.000043	22
H_3	0.000134	670
H_4	0.00007	350

the hash functions mentioned in Section 3.3 as the hash functions map a point in the curve to a random string or map a random string to a point in the curve or to a value which is co-prime to the prime characteristic of the curve. Overall, the implementation takes only 236 kB of memory on the Edison board. In Table III, the execution time and clock cycles required for Tate pairing and the four hash functions in Edison board have been enlisted.

6. DISCUSSION AND CONCLUSIONS

In this paper we have developed and described in detail a lightweight protocol for authentication, key exchange and secure message communication, applicable to hierarchical architectures in the IoT framework. We have used PUFs and cryptographic pairing for making our system resistant against active and passive adversarial attacks. We have also provided the formal proofs of security of the proposed protocol under the SK security and UC framework. The proposed protocol avoids the shortcomings of several previously proposed related PUF based authentication protocols, and is thus robust against reported attacks on such protocols. We have assumed that the CRP database is stored in a secure server, and even if the adversary accesses the database for a limited time, she would not be able to retrieve the private keys of the legitimate users, and also would be unable to decrypt the data transmitted over the network. However, there is a possibility that the compromised server can make a legitimate device communicate with an illegitimate or malicious node. In our future work, we will try to evade this problem to make the protocol more secure and robust.

From the implementation perspective, we have also presented low-overhead hardware/software co-design of the proposed architecture on an IoT test bed.Our future work would be directed towards optimisation of cryptographic pairing to further reduce the memory footprint, analysis of power consumption required to execute the protocol and evaluation of the robustness of the implementation with respect to implementation-specific attacks such as side channel analysis.

REFERENCES

MIRACL Crypto SDK. `http://www.miracl.com/miracl-sdk`.

ACAR, T., LAUTER, K. E., NAEHRIG, M., AND SHUMOW, D. 2011. Affine pairings on ARM. *IACR Cryptology ePrint Archive 2011*, 243.

AVOINE, G., CARPENT, X., AND HERNANDEZ-CASTRO, J. 2015. Pitfalls in ultralightweight authentication protocol designs. *Mobile Computing, IEEE Transactions on PP,* 99, 1–1.

AYSU, A., GULCAN, E., MORIYAMA, D., SCHAUMONT, P., AND YUNG, M. 2015. End-to-end design of a puf-based privacy preserving authentication protocol. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*. 556–576.

BATINA, L., GUAJARDO, J., KERINS, T., MENTENS, N., TUYLS, P., AND VERBAUWHEDE, I. 2006. An elliptic curve processor suitable for rfid-tags. *IACR Cryptology ePrint Archive 2006*, 227.

BATINA, L., MENTENS, N., SAKIYAMA, K., PRENEEL, B., AND VERBAUWHEDE, I. 2006. Low-cost elliptic curve cryptography for wireless sensor networks. In *Security and Privacy in Ad-Hoc and Sensor Networks, Third European Workshop, ESAS 2006, Hamburg, Germany, September 20-21, 2006, Revised Selected Papers*. 6–17.

BONEH, D. AND FRANKLIN, M. K. 2003. Identity-based encryption from the weil pairing. *SIAM J. Comput. 32,* 3, 586–615.

BRZUSKA, C., FISCHLIN, M., SCHRÖDER, H., AND KATZENBEISSER, S. 2011. Physically uncloneable functions in the universal composition framework. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*. 51–70.

CANETTI, R. 2001. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. 136–145.

CANETTI, R. AND KRAWCZYK, H. 2001. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*. 453–474.

CHEN, W. 2012. An ibe-based security scheme on internet of things. In *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*. Vol. 3. IEEE, 1046–1049.

DELVAUX, J., GU, D., SCHELLEKENS, D., AND VERBAUWHEDE, I. 2014. Secure lightweight entity authentication with strong pufs: Mission impossible? In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*. 451–475.

DUBEY, G., KHURANA, V., AND SACHDEVA, S. 2015. Implementing security technique on generic database. In *Eighth International Conference on Contemporary Computing, IC3 2015, Noida, India, August 20-22, 2015*. 370–376.

GASSEND, B., CLARKE, D. E., VAN DIJK, M., AND DEVADAS, S. 2002. Controlled physical random functions. In *18th Annual Computer Security Applications Conference (ACSAC 2002), 9-13 December 2002, Las Vegas, NV, USA*. 149–160.

GREWAL, G., AZARDERAKHSH, R., LONGA, P., HU, S., AND JAO, D. 2012. Efficient implementation of bilinear pairings on ARM processors. In *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*. 149–165.

GUO, Z. AND XU, L. 2015. Research of security structure model for web application systems based on the relational database. *IJSN 10,* 4, 207–213.

HELFMEIER, C., BOIT, C., NEDOSPASOV, D., AND SEIFERT, J.-P. 2013. Cloning physically unclonable functions. In *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on*. IEEE, 1–6.

HERREWEGE, A. V., KATZENBEISSER, S., MAES, R., PEETERS, R., SADEGHI, A., VERBAUWHEDE, I., AND WACHSMANN, C. 2012. Reverse fuzzy extractors: Enabling lightweight mutual authentication for puf-enabled rfids. In *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*. 374–389.

KATZ, J. AND LINDELL, Y. 2007. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press.

KATZENBEISSER, S., KOÇABAS, Ü., VAN DER LEEST, V., SADEGHI, A., SCHRIJEN, G. J., AND WACHSMANN, C. 2011. Recyclable pufs: logically reconfigurable pufs. *J. Cryptographic Engineering 1,* 3, 177–186.

KIM, M., LEE, H. T., LING, S., REN, S. Q., TAN, B. H. M., AND WANG, H. 2016. Better security for queries on encrypted databases. *IACR Cryptology ePrint Archive 2016*, 470.

KOÇABAS, Ü., PETER, A., KATZENBEISSER, S., AND SADEGHI, A. 2012. Converse puf-based authentication. In *Trust and Trustworthy Computing - 5th International Conference, TRUST 2012, Vienna, Austria, June 13-15, 2012. Proceedings*. 142–158.

KUMAR, S., GUAJARDO, J., MAES, R., SCHRIJEN, G.-J., AND TUYLS, P. 2008. Extended Abstract: The Butterfly PUF Protecting IP on every FPGA. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust*. HOST '08. 67–70.

LEE, Y. K., SAKIYAMA, K., BATINA, L., AND VERBAUWHEDE, I. 2008. Elliptic-curve-based security processor for RFID. *IEEE Trans. Computers 57,* 11, 1514–1527.

LIM, D. 2004. Extracting Secret keys from Integrated Circuits.

MAES, R., TUYLS, P., AND VERBAUWHEDE, I. 2009. Low-overhead implementation of a soft decision helper data algorithm for SRAM pufs. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*. 332–347.

MAJZOOBI, M. AND KOUSHANFAR, F. 2011. Time-bounded authentication of fpgas. *IEEE Transactions on Information Forensics and Security 6,* 3-2, 1123–1135.

MAJZOOBI, M., KOUSHANFAR, F., AND POTKONJAK, M. 2008. Lightweight secure pufs. In *2008 International Conference on Computer-Aided Design, ICCAD 2008, San Jose, CA, USA, November 10-13, 2008*. 670–673.

MAJZOOBI, M., ROSTAMI, M., KOUSHANFAR, F., WALLACH, D. S., AND DEVADAS, S. 2012. Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching. In *2012 IEEE Symposium on Security and Privacy Workshops, San Francisco, CA, USA, May 24-25, 2012*. 33–44.

MASS, M. 2004. Pairing-Based Cryptography.

MURTHY, C. S. R. AND MANOJ, B. 2004. *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

MUTTI, S., BACIS, E., AND PARABOSCHI, S. 2015. Sesqlite: Security enhanced sqlite: Mandatory access control for android databases. In *Proceedings of the 31st Annual Computer Security Applications Conference, Los Angeles, CA, USA, December 7-11, 2015*. 411–420.

OSTROVSKY, R., SCAFURO, A., VISCONTI, I., AND WADIA, A. 2013. Universally composable secure computation with (malicious) physically uncloneable functions. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. 702–718.

ÖZTÜRK, E., HAMMOURI, G., AND SUNAR, B. 2008. Towards robust low cost authentication for pervasive devices. In *Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2008), 17-21 March 2008, Hong Kong*. 170–178.

PAPPU, R. 2001. Physical One-way functions. Ph.D. thesis, Massachusetts Institute of Technology, USA.

RÜHRMAIR, U. 2012. Simpl systems as a keyless cryptographic and security primitive. In *Cryptography and Security*. Springer, 329–354.

RÜHRMAIR, U., BUSCH, H., AND KATZENBEISSER, S. 2010. Strong pufs: Models, constructions, and security proofs. In *Towards Hardware-Intrinsic Security - Foundations and Practice*. 79–96.

RÜHRMAIR, U., SEHNKE, F., SÖLTER, J., DROR, G., DEVADAS, S., AND SCHMIDHUBER, J. 2010. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*. 237–249.

RUHRMAIR, U. AND VAN DIJK, M. 2013. Pufs in security protocols: Attack models and security evaluations. In *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 286–300.

SHIRASE, M., MIYAZAKI, Y., TAKAGI, T., HAN, D., AND CHOI, D. 2009. Efficient implementation of pairing-based cryptography on a sensor node. *IEICE Transactions 92-D,* 5, 909–917.

SILVERMAN, J. H. 1986. *The arithmetic of elliptic curves*. Graduate texts in mathematics. Springer, New York, Berlin. 2e tirage corrig 1992.

XIONG, X., WONG, D. S., AND DENG, X. 2010. Tinypairing: A fast and lightweight pairing-based cryptographic library for wireless sensor networks. In *2010 IEEE Wireless Communications and Networking Conference, WCNC 2010, Proceedings, Sydney, Australia, 18-21 April 2010*. 1–6.

YANG, L., YU, P., BAILING, W., XUEFENG, B., XINLING, Y., AND GENG, L. 2013. Iot secure transmission based on integration of ibe and pki/ca. *International Journal of Control and Automation 6,* 2, 245–254.

YOSHITOMI, M., TAKAGI, T., KIYOMOTO, S., AND TANAKA, T. 2008. Efficient implementation of the pairing on mobilephones using BREW. *IEICE Transactions 91-D,* 5, 1330–1337.

YU, M. M., M'RAÏHI, D., SOWELL, R., AND DEVADAS, S. 2011. Lightweight and secure PUF key storage using limits of machine learning. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*. 358–373.