Building PUF Based Authentication and Key Exchange Protocol for IoT Without Explicit CRPs in Verifier Database

Urbi Chatterjee, Vidya Govindan, Rajat Sadhukhan, Debdeep Mukhopadhyay, Rajat Subhra Chakraborty, *Senior Member, IEEE*, Debashis Mahata, and Mukesh M. Prabhu

Abstract—Physically Unclonable Functions (PUFs) promise to be a critical hardware primitive to provide unique identities to billions of connected devices in Internet of Things (IoTs). In traditional authentication protocols a user presents a set of credentials with an accompanying proof such as password or digital certificate. However, IoTs need more evolved methods as these classical techniques suffer from the pressing problems of password dependency and inability to bind access requests to the "things" from which they originate. Additionally, the protocols need to be lightweight and heterogeneous. Although PUFs seem promising to develop such mechanism, it puts forward an open problem of how to develop such mechanism without needing to store the secret challenge-response pair (CRP) explicitly at the verifier end. In this paper, we develop an authentication and key exchange protocol by combining the ideas of Identity based Encryption (IBE), PUFs and Key-ed Hash Function to show that this combination can help to do away with this requirement. The security of the protocol is proved formally under the Session Key Security and the Universal Composability Framework. A prototype of the protocol has been implemented to realize a secured video surveillance camera using a combination of an Intel Edison board, with a Digilent Nexys-4 FPGA board consisting of an Artix-7 FPGA, together serving as the IoT node. We show, though the stand-alone video camera can be subjected to man-in-the-middle attack via IP-spoofing using standard network penetration tools, the camera augmented with the proposed protocol resists such attacks and it suits aptly in an IoT infrastructure making the protocol deployable for the industry.

Index Terms—Physically unclonable functions, elliptic curve cryptography, identity based encryption, internet of things, device authentication, key management

1 Introduction

IoTs have opened up an ubiquitous sensing-communicating-actuating network with information sharing across platforms, blended seamlessly in various areas of modern day-to-day living. But as with most emerging technologies, innovation comes first, and security is only an afterthought in reaction to discovered vulnerabilities. The devices deployed in an IoT framework usually generate large quantities of security-sensitive data. One of the major security challenges in IoT framework is the authentication and key management of potentially billions of devices deployed in the network. We try to address this problem and provide a lightweight and secure solution using PUFs and IBE [1]. A PUF circuit realization can be thought to be an unconventional, lightweight hardware security primitive [2] proposed in various security applications such as IC anti-counterfeiting, device identification and authentication, binding hardware to software platforms, secure storage of cryptographic secrets, keyless secure communication etc. A Silicon PUF [3] is a mapping $\gamma : \{0,1\}^n \rightarrow \{0,1\}^m$, where the output m-bit "response" are unambiguously identified by both the n "challenge" bits and the *unclonable, instance-specific* system behaviour. So, it can act as a hardware *fingerprint generator* for the IC in which it is included. We can adopt this property to *uniquely identify* each devices in the IoT framework. A specific challenge and its corresponding response together form a *Challenge-Response Pair* (CRP) for a given PUF instance. PUF based authentication protocols rely on this "challenge-response authentication" mechanism, rather than on a single secret cryptographic key. The response generated on-the-fly by the challenge applied to a PUF instance can be used to generate session key for secure message encryption; thus *offloads the complexity of managing and storing the keys* for IoT device. We make following contribution in this paper:

- We propose an authentication and key exchange protocol combining the concepts of PUF, IBE and Key-ed Hash Function. The protocol solves an open problem in the domain of PUF based protocols, *alleviating the overhead from the verifier to store the CRP database of the PUF and the dependency of imposing security mechanism*

- *U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, and R.S. Chakraborty are with the Secure Embedded Architecture Laboratory (SEAL), Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India. E-mail: {urbi.chatterjee, debdeep, rschakraborty}@cse.iitkgp.ernet.in, {vidya.govindan, rajat.sadhukhan}@iitkgp.ac.in.*
- *D. Mahata and M.M. Prabhu are with the Wipro Technogies, Bengaluru, Karnataka 560 035, India. E-mail: {debashis.mahata, mukesh.prabhu}@wipro.com.*

Manuscript received 31 July 2017; revised 7 Apr. 2018; accepted 17 Apr. 2018. Date of publication 1 May 2018; date of current version 10 May 2019. (Corresponding author: Urbi Chatterjee.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2018.2832201

to keep it secret. In traditional PUF based protocols, if a verifier needs to authenticate k IoT nodes, let us assume that it stores l number of m-bit challenges and its corresponding n bit responses. Then the space complexity is: $\mathcal{O}((m+n) \times l \times k)$. Now, if we consider the IoT framework, the "smart" devices (prover and verifier) are very resource constrained and more susceptible to be a target for active and passive attacks. In many cases prover device is just a sensor node and verifier device are mobile, bridge or router. Accessing CRP database by the smart devices itself is a security risk because smart devices are easy target for attacker. In order to offload storage requirement from verifier and to eliminate risk of getting CRP database compromised, we stores just a single key in the NVM of verifier for authentication of all prover nodes under it using a key-ed hash function (space complexity: constant). This way it would be easier to protect a single key instead of securing a whole CRP database. Additionally instead of using CRP database directly we generate a new security association information between prover and verifier that hides the correlation between the challenge and response of the PUF and can be stored as public information.

- We prove formally the security of the protocol in the Session Key Security model and the Universal Composability framework [4].
- We implement a prototype of the protocol to securely authenticate a video surveillance camera, commercially purchased and devoid of any inbuilt security feature. The prototype was implemented following a hardware/software co-design, by connecting the camera to an Intel Edison board, providing the IP and hosting the protocol operation, while the hardware circuit of the PUF is implemented and unique ID is generated from a Artix-7 FPGA. But, PUF responses are corrupted by noise and other environmental factors when deployed in an embedded system. Hence helper data algorithm or fuzzy extractor [5] is used to generate cryptographic keys with appropriate entropy from noisy and non-uniform random PUF responses. To perform this task, we design a BCH encoder circuit to generate the helper data from the responses of the PUF. This helper data along with a BCH decoder can then be used to re-generate the correct response from the actual response of a PUF for a specific challenge. It is to be noted that the BCH encoder and decoder circuit are implemented in Artix-7 FPGA.
- We first show a man-in-the-middle attack on the commercial video camera, and then show when the proposed protocol is enabled, the attack is subverted. We show that the protocol is lightweight, consumes low power, and has a low latency, suiting the requirements of IoT.

The rest of the paper is organized as follows. In Sections 2 and 3, we provide the security assumptions and the background of the work. In Section 4, we present our proposed authentication and key exchange protocol. The correctness and security analyses of the proposed scheme are described in Section 5. The experimental setup, attack scenario and resource overhead results have been provided in Section 6. We conclude the paper with future research directions in Section 7.

2 System Assumptions and Goals

System Model. The setting assumed is that each IoT node tries to authenticate to a verifier and communicate with the verifier or with another node. Each node is enabled with a PUF and has the capability to perform two elliptic curve operations, namely scalar multiplication and a pairing operation along with a standard cryptographic hash function. On the other hand the verifier is assumed to have the capability to compute keyed hash function, where the key is stored in a non-volatile memory.

Threat Assumptions. We assume the adversary can have access to the communication channel and can not only be a passive observer, but can tamper the channel with malicious data as an active adversary. The goal of the adversary is to authenticate to the verifier on behalf of the legitimate nodes, without possession of the node. For a PUF instance embedded in an IoT node, its challenge-response characteristics is an implicit property, and is thus not accessible to the adversary. Further, the attacker can corrupt the verifier (as by a malware) and can obtain access to the databases which the verifier possesses. However, we assume that the attacker cannot gain knowledge of the secret key stored on the verifier.

Attack Models. To formally proof the security of the protocol, we introduce two models which we will briefly discuss here.

- *Session Key Security Model*: Here all parties involved in the protocol are assumed to be 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 5.1.3, 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. We have shown three different scenarios where: a) The verifier and the two communicating parties are honest (ideal case), b) The verifier is corrupt, c) Either of the two communicating parties or both are corrupt. In real life implementation, we can picture *case (b) and (c)* as the attacker can control the internal functioning of the party and tries to send some malicious information to disrupt the system.

In this work, we do not address the subsequent encryption of the messages between the nodes, but sketch that the public-private key pair established can be used to communicate using established protocols [1].

Design Goals. Next, we briefly discuss the design goals of the proposed PUF based Authentication and Key Exchange Protocol:

- *No explicit key storage in 'Things':* Instead of having explicit key storage, a PUF instance will be embedded in each IoT data node to provide unique identity to the device.
- *Lightweight and minimal overhead on execution time :* The hardware overhead, power-consumption of the PUF enabled node and the latency to authenticate a legitimate node should be very less.
- *No explicit storage of raw CRP with verifier and model building resistance:* The verifier will not have access to the raw CRP database of the PUF of the prover node. This is to ensure that if the verifier gets compromised, no one should be able to mathematically clone the PUF instances using the CRP databases.
- *Unlimited authentications:* The protocol will have moderate input-output space and can have unlimited authentication rounds repeating same challenges.
- *Security association mapping for CRPs:* A mapping is done between the challenge and response for each entries in the CRP database so that it can be stored publicly in a resourceful device ensuring its integrity. The verifier can access it at the time of authentication without any advantages to the attacker.
- *Efficient management of public/ private keys without central authority:* There is no need to involve central certificate authority to sign the public keys. A verifier can easily verify the public key of the prover as it holds information derived from the PUF instance of the prover. The public-private key should be suitably tied to the PUF instance of the node, and that serves as the *root of trust*.

3 Alternative Approaches and Related Work

In this section we discuss conventional protocols and their shortcomings for authentication and key exchange among the nodes of an IoT system.

3.1 PUF Based Protocols

Several lightweight PUF-based authentication protocols [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] have been proposed in the past. But in[17], the authors demonstrated several vulnerabilities such as Denial-of-Service (DoS) attack, synchronization problem, replay attack, token/ server impersonation, modelling attack, lack of integrity checking of the helper data, compromise of code and data at runtime, limited local authentication, single point of failure that have made these protocols unacceptable in their original form. The mutual authentication protocol proposed in [18] has considerable hardware overhead; hence, it is not suitable for resource-constrained devices. Moreover, in most of the PUF based authentication schemes, a verifier node granting authenticity to a prover node, has prior access to a subset of CRP database or a model of the PUF instance embedded at the prover node. Now, if we map this setup in a hierarchical network of IoT framework, it may expand the attack surface substantially, as the integrity of CRP details at lower level nodes may get compromised due to easy accessibility. Hence, we cannot adopt any of these protocols in its current form.

In this paper, we have tried to overcome the above-mentioned problems. In our scheme, the prover (resource-constrained) node is PUF-enabled, but the verifier (less resource-constrained) node does not need to hold the subset of the CRP database or the model of the PUF instance. Rather, it contains a keyed hash function which is used to authenticate the PUF instance without knowing the actual response of a given challenge. We have assumed that the key is stored in a secure non-volatile memory. However, the prover does not need to explicitly store any key, rather the secret is generated from the response of a PUF which is embedded in the device.

3.2 Public Key Based Protocols

Authentication and key exchange have been traditionally handled by the use of public key encryption. The two conventional ways of handling encryption is by the use of Public Key Infrastructure (PKI) or by the use of Identity Based Encryptions (IBE). In [19], new protocols have been proposed for the IP protection problem on FPGAs using PUFs and PKI based public key cryptography. But PKI has been plagued with several shortcomings of non-uniform standards, and most importantly the difficulty of handling certificates generated by a trusted third party, virtually making it infeasible for IoT applications where billions of devices are expected to communicate. As an alternative, identity based encryptions are attractive as they provide a mechanism for generating public keys from publicly known information. However, in classic IBE the secret keys of a node are not tied to its physical identity, and the proof of identity is usually in the form of a password or a digital certificate that include a user's public key. Moreover, some of these secrets need to be explicitly stored in the nodes. Further, classic IBE requires a Public Key Generator (PKG) which is used to generate private keys for the nodes and transfer through secured channels. This makes the key exchange unwieldy and difficult for real life deployment for the scalability of IoT applications. In [20], Wallrabenstein has proposed to use PUF based Elliptic Curve Cryptosystem for IOT framework, but storing helper data for each challenge in the node can lead to unacceptable memory overhead in resource-constrained devices.

In the proposed protocol, we have blended IBE with identity generated by the PUF embedded in a node. It leads to a certificate-less protocol, where no explicit keys need to be stored in the nodes, while IBE provides security based on strong well-founded hard problems. The key exchange in the proposed protocol is made seamless by allowing the nodes with the PUFs generating its keys, while the verifier simply checks its authenticity and passes a verified public key to another node for further communications.

Security of Commercial IoT Appliances. Surprisingly, even with the growing importance of security, several IoT appliances have very little to no support for it. As a use-case, in this paper we study video surveillance cameras, which are considered as a very popular IoT application. Till now, several passive and active attacks [21], [22] such as visual layer attacks, abusing covert channel and data ex-filtration attacks, jamming, Denial-of-service, and side channel attacks have been proposed for video surveillance system. As a countermeasure, many public key infrastructure based user authentication protocols[23], [24] were proposed in

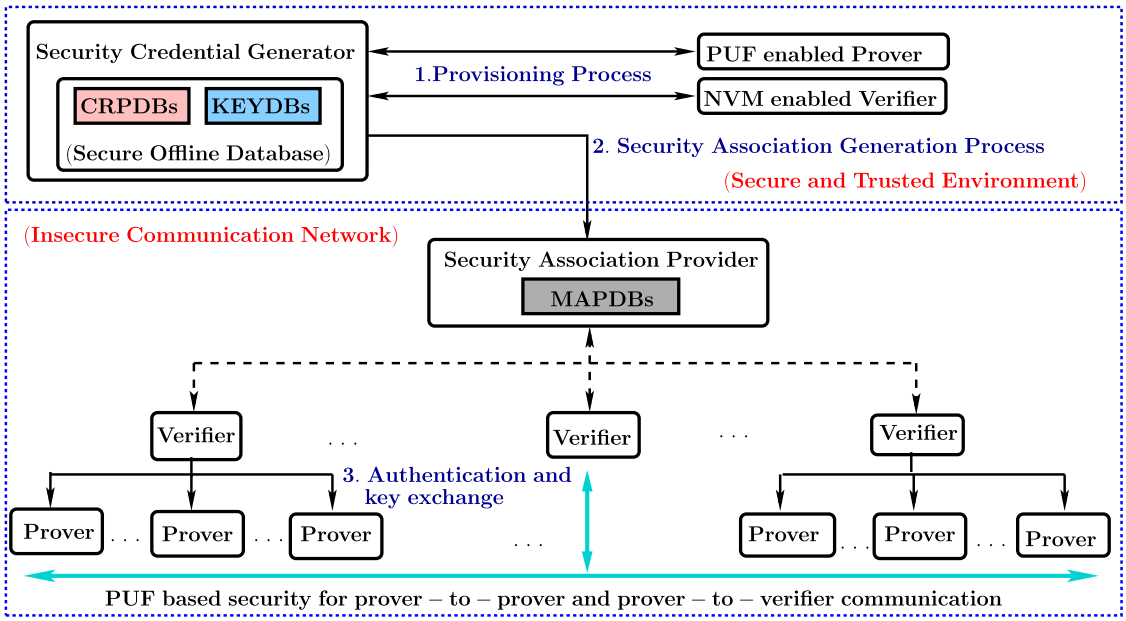

Fig. 1. Hierarchical IoT architecture and proposed secure communication mechanism.

literature. However the fact remains that many network-enabled camera vendors do not use data encryption, to increase the throughput and to decrease memory and power footprint. Additionally, some of the current video streaming protocols such as RTP, RTSP and video steaming engines such as WOWZA, Mjpg-Streamer etc. do not even support secure network protocols such as SSL. This inspires us to develop PUF based authentication and key exchange protocol which will ensure the device authentication *irrespective of the security level of the network protocol running on it*.

4 Proposed Authentication and Key Exchange Protocol

In this section, we describe the authentication and key exchange protocol that can be suitably implemented in an IoT infrastructure. Fig. 1 represents the functional blocks of the proposed security architecture. The architecture consists of four major components: the Security Credential Generator (SCG), the Security Association Provider (SAP), the Verifier Node and the IoT Node. The IoT nodes, which play the role of *prover*, reside at the lowest level of the architecture. In our proposal, we assume these IoT nodes to be PUF-enabled, and having low hardware and software footprint and limited computational abilities. They prove their authenticity using respective embedded PUF instances to the immediate upper layer nodes, which play the role of *verifier* and are relatively resourceful.

The proposed protocol has two main phases, enrolment phase and authentication & key exchange phase. The *Enrolment phase* consists of two sub processes and executes in a secure and trusted environment. Once the manufacturing of the verifier and IoT prover nodes are done, the SCG executes a *Provisioning Process*. In this process, the characterization of the PUF instance is done for each of the IoT nodes and stored in CRP databases (CRPDBs). Similarly, a randomly selected secret key is assigned for each verifier and stored in the NVM of the verifier as well as the key databases (KEYDBs) (marked as '1'). To resist against modelling attack of the PUF instance, the CRPDBs and KEYDBs are assumed to be stored in a secure "offline" database in a trusted environment, outside the reach of the typical IoT "node-to-node" communication. *These database entries are never directly used for authentication*.Next, each verifier node is assigned to authenticate a set of IoT prover nodes. In *Security Association Generation Process* a security relationship mapping between IoT node and verifier node is created (marked as '2') using each CRP entry of the prover node, the secret key associated with its corresponding verifier node and some randomly selected entities by the SCG. It hides the challenge-response correlation of the PUF instance. These mapping entries are stored in Mapping Databases (MAPDBs) in the SAP maintained outside the trusted environment. MAPDBs are generated in such a way that access to this database would not help the adversary to model the PUF instance, and the integrity of the entries are maintained so that *the trusted party can verify any illegitimate modification during the protocol execution*.In the *Authentication and key exchange phase*, the verifier uses challenges randomly selected from MAPDBs and validates responses from the prover node dynamically at the time of protocol execution. The protocol is designed in a way that *both the prover and the verifier mutually authenticate each other*. Finally, the verifier node coordinates among different prover nodes for generation and sharing of public keys (marked as '3').

4.1 Public Mathematical Parameters

Our scheme requires that the communicating parties must agree on some mathematical parameters before initiating communication. For some large prime value q, we define an elliptic curve and generate three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ on the points of an elliptic curve to define cryptographic pairing. Pairing is an admissible bilinear map e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ which satisfies the following three properties:

1) *Bilinearity*: $\forall a, b \in F_q^*, \forall P \in \mathbb{G}_1, Q \in \mathbb{G}_2 : e(aP, bQ) = e(P, Q)^{ab}$.

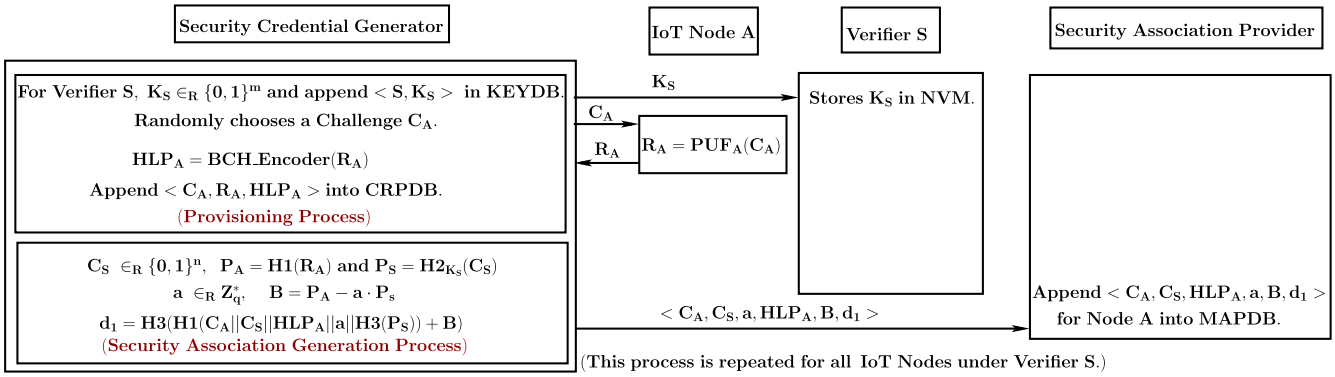

Fig. 2. Enrolment phase of the proposed protocol.

2) *Non-degeneracy*: $e(P, Q) \neq 1$.
3) *Computability*: There exist an efficient algorithm to compute e.

For further details, please refer to Section 2 of [25]. We also need to choose three secure cryptographic hash functions: $H1$: $\{0,1\}^n \rightarrow \mathbb{G}_1^*$, $H2$: $\{0,1\}^n \times \{0,1\}^m \rightarrow \mathbb{G}_2^*$, $H3$: $\mathbb{G}_2 \rightarrow \{0,1\}^n$, where n and m are the bit lengths of the PUF response and secret key, respectively, in our context. So, the public mathematical parameters are: $< q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, n, H_1, H_2, H_3 >$.

4.2 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. 2, and are summarized as follows:

- In the provisioning process, the SCG first randomly selects an m-bit key K_S and assigns it to the NVM of Verifier S. It also stores K_S in the KEYDBs.
- Then it sends a random challenge C_A to the IoT Node A. Node A applies the challenge C_A to its PUF, and generates the output $R_A = PUF(C_A)$, and returns it to the SCG.
- The SCG generates the helper data $HLP_A = BCH\_Encoder(R_A)$ and stores it along with the challenge and response by appending $< C_A, R_A, HLP_A >$ to its CRPDBs.
- Next, in Security Association Generation Process, the SCG randomly generates an n-bit challenge C_S, and then it calculates

$$P_S = H2_{K_S}(C_S), P_A = H1(R_A).$$

Then, the SCG randomly selects an element a from Z_q^* and calculates

$$B = P_A - a \cdot P_S,$$
$$d_1 = H3(H1(C_A||C_S||HLP_A||a||H3(P_S)) + B).$$

Please note that $Z_q^* \stackrel{\text{def}}{=} \{x \in Z_q : gcd(x, q) = 1\}$ i.e., elements of Z_q with multiplicative inverses. In this way, a new tuple $< C_A, C_S, HLP_A, a, B, d_1 >$ is generated and stored in the MAPDBs of the *Security Association Provider*. This procedure is repeated according to the memory capacity of the SAP and the SCG and for all IoT nodes under Verifier S.

At the end of the enrolment phase for a given node A, the Verifier S supervising it will have only the secret key. For authentication, the SAP will transfer an entry randomly from the mapping database of the node A to the Verifier S. The Verifier S will calculate the response of the PUF on-the-fly to authenticate node A. Here, we have assumed that the Verifier S will securely store the secret key for the keyed hash function in a non-volatile memory. We can achieve this goal using the commercially available tamper-proof NVM chips, e.g., those used in *Trusted Platform Module* (TPM) [26].

4.3 The Authentication and Key Exchange Phase

The second phase of this protocol performing authentication and key sharing is described below as shown in Fig. 3. Consider a situation where IoT node A wishes to communicate with IoT node B, with both A and B being at the lowest levels of the hierarchy.

- At first, IoT node A initiates a request $< ID_A, ID_B >$ (i.e., the public identifiers of the two communicating nodes) to Verifier S for authentication. Verifier S forwards the request to the SAP.
- The SAP randomly chooses an entry $< C_A, C_S, HLP_A, a, B, d_1 >$ from $MAPDB_{AS}$ and sends it back to the Verifier S.
- Now, the Verifier S performs the following computations:

$$P_S = H2_{K_S}(C_S).$$

- If $d_1 == H3(H1(C_A||C_S||HLP_A||a||H3(P_S)) + B)$, then it calculates

$$P_A = a \cdot P_S + B.$$

It can be noted that if the Verifier S gets compromised, it can impersonate as Node A. To avoid this scenario, we suggest to use Strong PUFs and the mapping entry used for an authentication round is deleted from $MAPDB_{AS}$.
- Next, the Verifier S randomly chooses a value x such that $x \in_R Z_q^*$ and computes

$$Q_A = P_A + x \cdot P_S + H1(ID_A||ID_B),$$
$$V_A = e(P_A, x \cdot P_S),$$

and sends this value to node A as the tuple $< ID_B, C_A, HLP_A, Q_A >$. Please note that the nonce

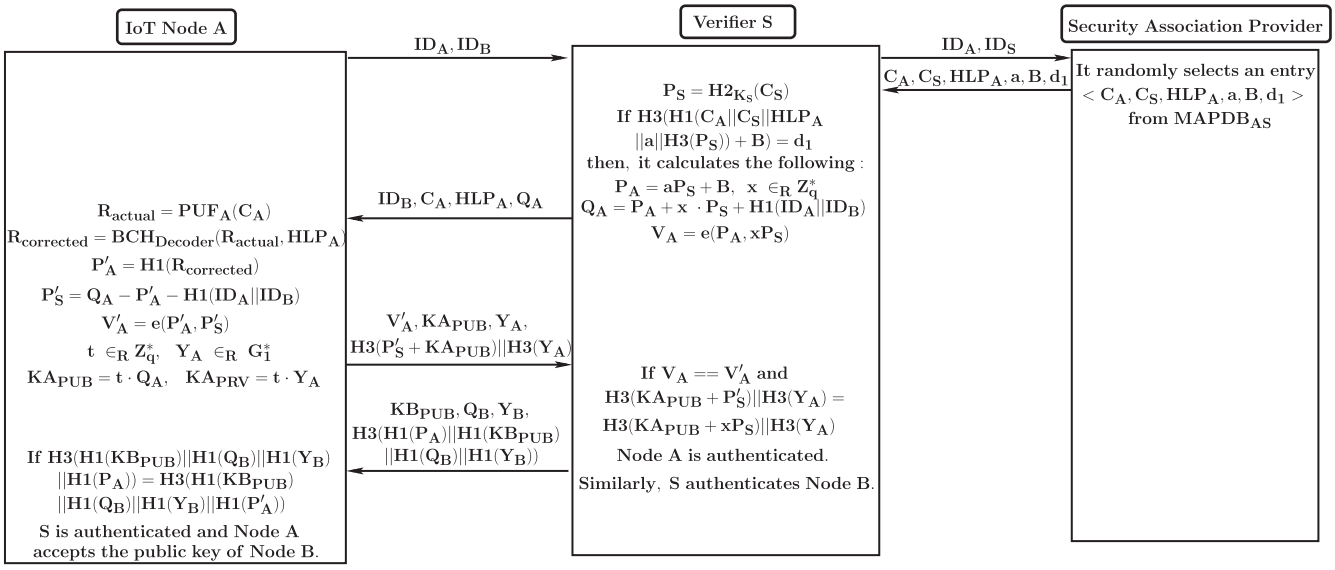

Fig. 3. The authentication and key exchange phase.

x is used to resist the replay attack and also acts as a timestamp for that specific instance of the protocol run. Generally it is very hard to mitigate DoS attacks at protocol level [27]. But in proposed protocol, we took two approaches from the verifier and the node's perspective. As the authentication request initiation is done by the node, it can keep track of exactly how many requests have been sent by it. In case, it is flooded with challenge requests, then it can temporarily shut down the protocol execution and can opt for approaches such as *exponential back-off algorithms* which is used for network congestion avoidance. On the verifier side, the timestamp x is used to keep track that currently which nodes are executing the protocol. Hence if new requests come for the same pair, it can immediately rejects them. This way the frequency of each request type can be limited.

- On receiving the message, node A first applies C_A to its PUF instance PUF_A and get the response R_{actual}.
- Next using helper data HLP_A and R_{actual} in $BCH\_Decoder$, it retrieves the corrected response $R_{corrected}$.
- It calculates the following:

$$P'_A = H1(R_{corrected}),$$
$$P'_S = Q_A - P'_A - H1(ID_A||ID_B),$$
$$V'_A = e(P'_A, P'_S).$$

- Next, node A randomly chooses two values t and Y_A such that $t \in_R Z_q^*$ and $Y_A \in_R \mathbb{G}_1^*$. Then it computes the public and private key pair

$$KA_{PUB} = t \cdot Q_A, KA_{PRV} = t \cdot Y_A,$$

and it sends the Verifier S the tuple $< V'_A, KA_{PUB}, Y_A, H3(P'_S + KA_{PUB})||H3(Y_A) >$.
- If V_A equals V'_A and $H3(P'_S + KA_{PUB})||H3(Y_A)$ equals $H3(x \cdot P_S + KA_{PUB})||H3(Y_A)$, the Verifier S accepts node A as an authenticated device, and accepts its public key.
- Since node A wishes to communicate with node B, it needs the Verifier S to authenticate node B. Hence, the Verifier S follows a similar procedure for node B as described above to authenticate node B, and accepts its public key KB_{PUB} upon successful authentication. Finally, it sends node A the tuple $< KB_{PUB}, Q_B, Y_B, H3(H1(P_A)||H1(KB_{PUB})||H1(Q_B)||H1(Y_B)) >$. On receiving it, if node A finds that $H3(H1(P_A)||H1(KB_{PUB})||H1(Q_B)||H1(Y_B))$ equals $H3(H1(P'_A)||H1(KB_{PUB})||H1(Q_B)||H1(Y_B))$, then the Verifier S is authenticated, as only the Verifier S can retrieve the value of P_A using P_S, and node A accepts the public key of node B.

5 Security Analysis

Next we will describe two different attack models in which we will analyze the security of the proposed authentication and key exchange protocol.

5.1 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.

We prove that our protocol is secure against UM, which in turn ensures that the protocol is secure against AM.

Consider the following experiment under UM: the attacker Λ chooses to attack a session under test, and let κ be the shared session key of the session. A random coin tossing is performed, with its result encoded as a bit b. If $b = 0$, the value κ is given to the attacker Λ, otherwise a random value r is chosen from the probability distribution of keys generated by the protocol. The attacker outputs a bit b' at the end.

Definition 1 (Session Key Secure (SK-Secure) Protocol). *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.

5.1.1 Security Assumptions

As mentioned previously, there are two security assumptions at the foundation of the secure communication protocol proposed. The first security assumption is the *physical and mathematical unclonability of PUFs by a polynomial-time algorithm*, which implies that it is computationally infeasible to construct the challenge-response mapping of an arbitrary PUF instance. Although most PUF variants are physically unclonable at the current state-of-the-art (a notable exception being the successful effort of SRAM PUF cloning reported in [28]), successful mathematical modeling ("model-building attacks") have been widely reported [29]. However, by choosing relatively secure PUF variants such as *Lightweight Secure PUF* or *XOR PUF* [29], we can avoid both physical and mathematical cloning in practice. This security assumption is formalized in the definitions below:

Definition 2 (Decisional Uniqueness Problem (DUP) for PUF). *Given an n-bit output of an arbitrary 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 3 (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*

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

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 $PUF_N(C)$. Another way of interpreting the Decisional Uniqueness Problem Assumption is that the ensemble of tuples of type (C, PUF_{Adv}, z) is computationally indistinguishable from the ensemble of tuples of type $(C, PUF_{Adv}, PUF_N(C))$.

The second important security assumption is the *computational infeasibility of the Elliptic Curve Discrete Logarithm Problem* (ECDLP):

Definition 4 (Elliptic Curve Discrete Logarithm Problem). *Let $E(K)$ be a discrete elliptic curve over a finite field K; let there exist points $P, Q \in E(K)$ such that $Q \in <P>$, where P is a* primitive point *(capable of generating any arbitrary point on $E(K)$), and $<P>$ denotes the set of points generated by adding P to itself k times, for some integer k. The ECDLP problem is to find the value of the scalar multiple k, given P and Q. ECDLP is considered computationally intractable at the current state-of-the-art for proper choices of the curve $E(K)$.*

5.1.2 Correctness Proof of the Proposed Scheme

We consider a setting with two parties, IoT node A and the Verifier S monitoring the authentication procedure of node A. We denote the protocol by π. Recall that node A and the verifier contain PUF instance PUF_A and a secure NVM storing K_S. Moreover, let $output_{nodeA,\pi}(ID_B, C_A, HLP_A, Q_A)$ and $output_{S,\pi}(C_A, C_S, a, B)$ denote the respective outputs of node A and the Verifier S. We assume that this output takes the form of an element of G_3^* that is supposed to be considered as the identity of node A, and should be shared by node A and the verifier. Hence

$$\begin{aligned} &output_{nodeA,\pi}(ID_B, C_A, HLP_A, Q_A) \\ &= e(H1(BCH\_Decoder(PUF_A(C_A), HLP_A)), \\ &Q_A - H1(BCH\_Decoder(PUF_A(C_A), HLP_A)) \\ &\quad - H1(ID_A || ID_B)), \end{aligned}$$

and

$$\begin{aligned} &output_{S,\pi}(C_A, C_S, a, B) \\ &= e(a \cdot H2_{K_S}(C_S) + B, x \cdot H2_{K_S}(C_S)). \end{aligned}$$

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

Definition 5 (Correctness of Protocol). *A protocol π for authentication and key exchange is denoted as* correct *if there exists a negligible function $negl(\cdot)$, such that for every possible value of n*

$$\Pr[output_{nodeA,\pi}(C_A, HLP_A, Q_A) \neq output_{S,\pi}(C_A, C_S, a, B)] \leqslant negl(n).$$

It can be observed that

$$\begin{aligned} &e(a \cdot H2_{K_S}(C_S) + B, x \cdot H2_{K_S}(C_S)) = e(P_A, x \cdot H2_{K_S}(C_S)) \\ &= e(H1(BCH\_Decoder(PUF_A(C_A), HLP_A)), Q_A - H1 \\ &(BCH\_Decoder(PUF_A(C_A), HLP_A)) - H1(ID_A || ID_B)). \end{aligned}$$

This means that node A and the verifier will output the same value, thereby proving the correctness of the scheme.

It may be noted that the rationale for the choices of the public and private keys are based on [1]. The exact

description of the encryption process is beyond the scope of the present work, but for the sake of completeness, we would like to sketch that for encryption. For a random string w, the node A can compute, a string $\lambda = e(KB_{PUB}, Y_B)^w = e(t \cdot Q_B, Y_B)^w = e(Q_B, Y_B)^{t \cdot w}$, which can be used to confide a message to be sent to node B. The encryptor sends a hint for w to node B, which is $w \cdot Q_B$. The decryptor node B using the hint and its private key can compute this string by calculating $e(w \cdot Q_B, KB_{PRV}) = e(w \cdot Q_B, t \cdot Y_B) = e(Q_B, Y_B)^{t \cdot w} = \lambda$. This explains briefly the choices for the public and private keys in the proposed key exchange protocol.

5.1.3 Security Proof of the Proposed Scheme

From the security perspective, an authentication and key exchange protocol is secure if the output V_A generated by node A and the verifier are identical, and no adversary can correctly guess V_A for the challenge $< C_A, C_S, HLP_A, a, B, d1 >$ and x chosen randomly. This has been formulated by giving an adversary the values $< C_A, C_S, HLP_A, a, B, d1 >$ from a protocol execution, and observing whether she can distinguish between V_A generated by node A, and the verifier, or a completely random element of G_3^*.

We would show that breaking the proposed protocol is at least as difficult as solving the Decisional Uniqueness Problem for PUFs, i.e., a successful attack on the proposed protocol implies a feasible solution to the Decisional Uniqueness Problem for PUFs. 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 $\mathbf{Auth_{adv,\pi}(n, \zeta, PUF_{Adv}, V_{A_0}, V_{A_1})}$ **:**

1) The adversary *Adv* is provided:
   a) A PUF instance PUF_{Adv} and $\zeta = < C_A, C_S, HLP_A, a, B, Q_A, ID_A, ID_B >$ where $Q_A = ((a \cdot H2_{K_S}(C_S) + B) + (x \cdot H2_{K_S}(C_S))) + H1(ID_A || ID_B)$.
   b) Two identities V_{A_0} and V_{A_1}, calculated based on a chosen random bit $b \in \{0, 1\}$

$$V_{A_b} = e(a \cdot H2_{K_S}(C_S) + B, x \cdot H2_{K_S}(C_S))$$
$$V_{A_{1-b}} = h \in_R G_3^*.$$

2) The adversary *Adv* will output a value b'. If $b = b'$, the adversary Adv succeeds in the experiment.

Next we prove the following theorem.

Theorem 1. *The authentication and key exchange protocol π is secure in the presence of eavesdropping adversaries if the Decisional Uniqueness Problem Assumption holds.*

Proof. To prove this, we 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

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

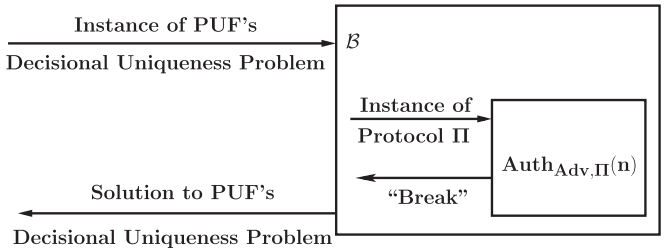

Fig. 4. The view of $\mathbf{Auth_{adv,\pi}}$ when it is run as a sub-routine of $\mathcal{B}$ ([30]).

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 Uniqueness problem. Now, the algorithm $\mathcal{B}$ takes as input a random Uniqueness Problem tuple (C_A, PUF_{Adv}, z_A) (where $z_A = PUF_A(C_A)$ or one random string belongs to $\{0, 1\}^*$) and proceeds as follows:

(1) *SetUp*: Provide *Adv* with PUF_{Adv}.
(2) *Input tuple*:
   (a) First it randomly chooses P_S and x.
   (b) It calculates $P_A = H1(z_A)$.
   (c) Then it calculates

$$Q_A = P_A + x \cdot P_S + H1(ID_A || ID_B).$$

   (d) Then sets $\zeta = < C_A, C_S, HLP_A, a, B, Q_A, ID_A, ID_B >$, which is perfectly random to the adversary *Adv*.
   (e) Next, it randomly chooses $b \in \{0, 1\}$.
   (f) It then calculates $V_{A_b} = e(P_A, x \cdot P_S)$ and $V_{A_{1-b}} = h \in_R G_3^*$
   (g) The algorithm $\mathcal{B}$ finally provides *Adv* the input tuple $< \zeta, V_{A_0}, V_{A_1} >$. If $z_A = PUF_A(C_A)$, then V_{A_b} will be equal to $e(P_A, x \cdot P_S)$ and it will be a valid input tuple. Otherwise, V_{A_0}, V_{A_1} both will be some random element of G_3^*.
(3) *Guess*: The adversary *Adv* returns b', a guess of b. If $b = b'$, then the algorithm $\mathcal{B}$ returns 1, implying that z_A are the correct responses of C_A. 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 of Uniqueness Problem, ε should be negligible (please refer to Fig. 4). Hence

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

□

Once the authentication is done successfully, node A selects a random value $t \in_R Z_q^*$. Then, it locally calculates {public,private} key pair $K1_{PUB} = t \cdot Q_A$ and $K1_{PRV} = t \cdot Y_A$. It keeps $K1_{PRV}$ secret and sends $K1_{PUB}$ to the verifier 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 Q_A 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 A. 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. From the result obtained in the previous theorem, we conclude that: *based on the complexity assumption of the Computational Discrete Log Problem, Decisional Uniqueness 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.*

Hence, DUP $\wedge$ ECDLP $\rightarrow \pi$ is SK-secure.

$$\Rightarrow \pi \text{ is not SK-secure} \rightarrow \neg \text{ DUP} \vee \neg \text{ ECDLP}.$$

Additionally, the protocol is designed in such a way that both prover and the verifier mutually authenticate each other. If an legitimate node A tries to impersonate as another legitimate node B under the same verifier using the same challenge set $< C_A, C_S >$, it will fail to do so. As the PUF's response does not depend on the value of K_S and it is only used regenerate the response, use of the same for two different nodes will not lead to *masquerading attack*. The proof is similar to that given above. Furthermore, SAP is a database holding the mapping entries. These entries (in MAPDB) are already encoded by Security Credential Generation process (refer to Section 4.2) and kept publicly. Only legitimate nodes can interpret information stored in a MAPDB entry. Hence SAP does not need to authenticate the verifier. Next, we prove the compatibility of the scheme with the universal composability framework.

5.2 Universal Composability Framework

The basic objective of *Universal Composability* (UC) 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 prove that the method of key exchange as proposed in this work is also compatible with similar composability properties. It follows the approach referred as "security by emulation of an ideal process". The primary concept of this principle is as given below [4]:

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 modeled 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.
6) A protocol π *securely realizes the task* f if π *emulates* the ideal process $\mathcal{F}$, 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.

5.2.1 UC Security of the Proposed Key Exchange Phase

The main concept of asymmetric key exchange ideal functionality $\mathcal{F}_{AKE}$ is that: if both the communicating parties are honest, the functionality provides them with two random identities, 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. If one of the communicating parties is corrupt, then the adversary can easily determine the identity of the corrupt party. $\mathcal{F}_{AKE}$ is parameterized by an integer N (the total number of permissible sessions), where a verifier runs with exactly t data nodes and the simulator $\mathcal{S}$. The working principle of $\mathcal{F}_{AKE}$ has been shown the Fig. 5. Next we prove the security of $\mathcal{F}_{AKE}$.

Theorem 2. *Protocol π securely realizes functionality $\mathcal{F}_{AKE}$.*

Proof. Here we assume that (a) the adversary possesses a PUF instance; (b) queries to the PUF are genuinely handed on to the simulator $\mathcal{S}$'s PUF, and (c) the PUF's answers are forwarded unmodified to the querying party throughout all the simulations. We consider different usage and security scenarios in turn.

Case-1: Verifier and Node A are Honest.

- *Setup:* Whenever the functionality $\mathcal{F}_{AKE}$ receives message (**establish-session$_{\mathbf{AKE}}$**, **sid**, **Node A**, **Verifier**) for the first time, the simulator $\mathcal{S}$ queries the PUF instance PUF_A for k random challenges $C_1, C_2, \ldots, C_k$, and obtains responses $R_{A1}, R_{A2}, \ldots, R_{Ak}$. Then, it creates a list $\mathcal{L}_A$ of k challenge-response pairs.
- It then hands over PUF_A to node A.
- On receiving a message (**establish-session$_{\mathbf{AKE}}$**, $\}$ sid, $Node\ A$, $Verifier$), $\mathcal{F}_{AKE}$ increments p by one and the simulator $\mathcal{S}$ sends ($deliver_{AKE}$, sid, $Verifier$) to $\mathcal{F}_{AKE}$.
- $\mathcal{F}_{AKE}$ then sends ($deliver_{AKE}$, sid, V_A, $Verifier$) to the verifier.
- Now the simulator $\mathcal{S}$ is activated again and it simulates that the verifier sends (ID_B, C_A, HLP_A, Q_A) to node A.
- When the adversary Λ instructs to send the latter message to node A, the simulator $\mathcal{S}$ sends ($deliver_{AKE}$, sid, V_A, $Node\ A$) to $\mathcal{F}_{AKE}$.

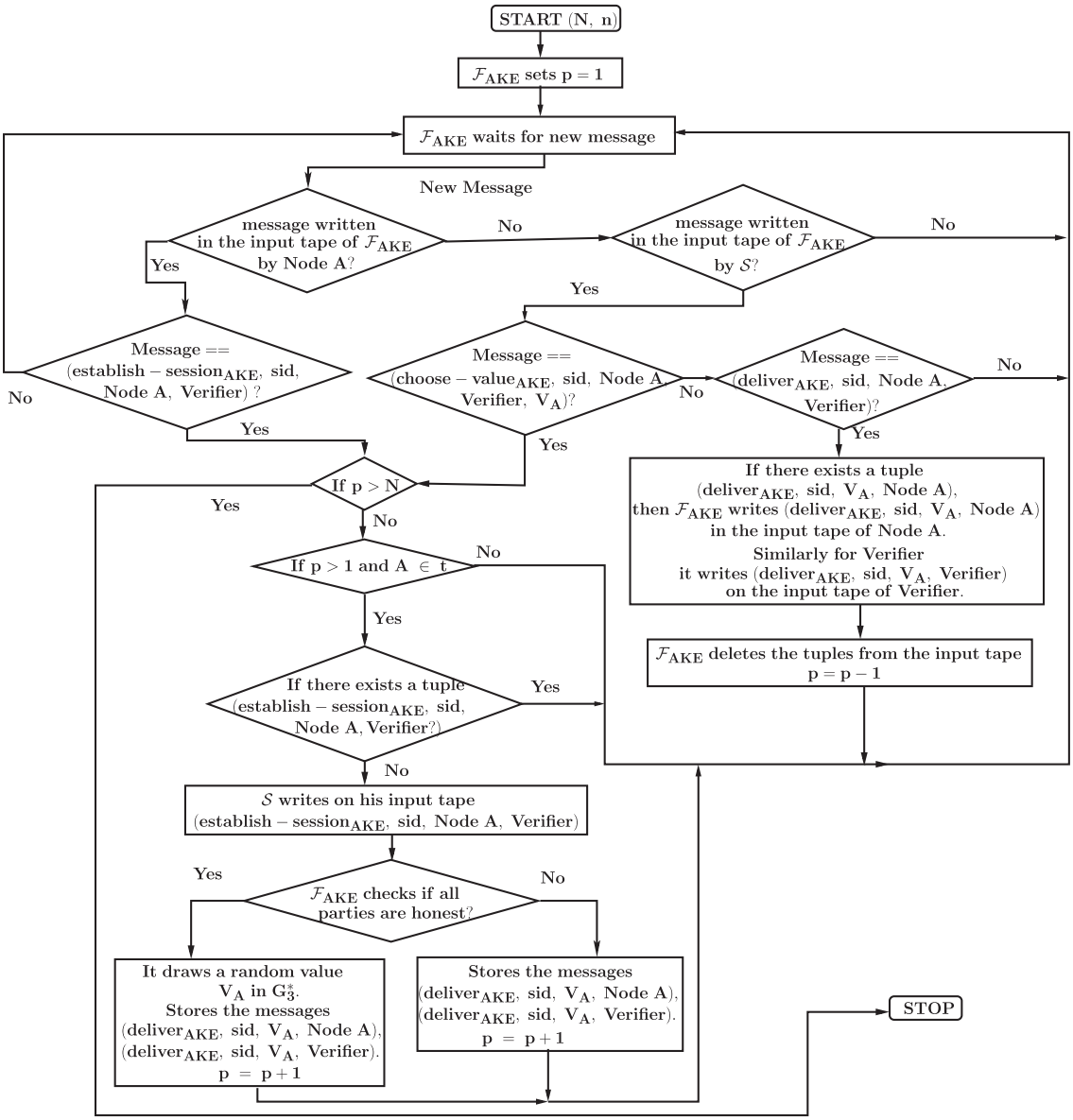

Fig. 5. The asymmetric key exchange ideal functionality.

- The probability that the value of $e(H1(BCH\_Decoder(PUF_A(C_A), HLP_A)), x \cdot H2_{K_S}(C_S))$ is equal to V_A is *negligible* (as proved in Section 5.1.3).

Case-2: Verifier is Corrupt.

- The simulator $\mathcal{S}$ let Verifier to instantiate PUF_A, and hands it over to node A.
- When the adversary Λ instructs to deliver message (ID_B, C_A, HLP_A, Q_A), then the simulator $\mathcal{S}$ can easily evaluate $V_A = e(a \cdot H2_{K_S}(C_S) + B, x \cdot H2_{K_S}(C_S))$, as the server is corrupt. But it is to be noted that $\mathcal{S}$ does not have access to K_S. It can only get the final value of V_A (refer to 5th point of Section 5.2).
- It next sends $(choose - value_{AKE}, sid, Node\ A, Verifier, V_A)$ to $\mathcal{F}_{AKE}$ as it has already calculated the value of V_A and $\mathcal{F}$ increments the value of p by one.
- Finally, $\mathcal{S}$ sends the messages $(deliver_{AKE}, sid, V_A, Node\ A)$ to $\mathcal{F}_{AKE}$.
- Hence in this case the ID provided by $\mathcal{F}$ and the identity calculated from the challenges given by the server is same.
- But node A later chooses a random value $t \in Z_q^*$ after getting the V_A, 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 A. 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. Now as node A randomly selects the value of t and Q_A, Y_A 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. 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.

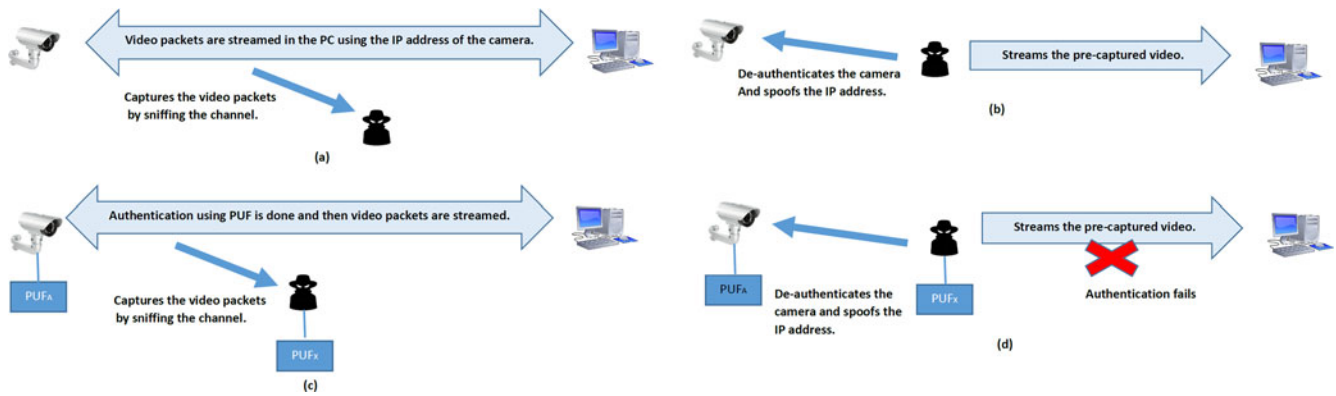

Fig. 6. Attack on video surveillance system and protection against it: (a) and (b) show the successful attack in the absence of PUF based authentication mechanism, while (c) and (d) show the prevention of the attack in the presence of the proposed PUF based authentication system.

Case-3: Node A is Corrupt.

This case covers the situation if a party willingly hands over its PUF to the adversary Λ. So, in this case, we show that the adversary Λ can easily retrieve the value of private key for that particular party.

- The set up phase is same as given in Case-1.
- On receiving a message $(establish-session_{AKE}, sid, Node\ A, Verifier)$, the simulator $\mathcal{S}$ increments p by one and sends $(choose-value_{AKE}, sid, Node\ A, Verifier, V_A)$ to $\mathcal{F}_{AKE}$.
- It is activated again and sends $(deliver_{AKE}, sid, Verifier)$ to $\mathcal{F}_{AKE}$.
- Verifier writes the V_A on its local tape and $\mathcal{S}$ is activated again.
- It simulates the verifier sending (ID_B, C_A, HLP_A, Q_A) node A.
- When the adversary Λ instructs to deliver the latter message to node A, $\mathcal{S}$ sends $(deliver_{AKE}, sid, V_A, Node\ A)$ to $\mathcal{F}_{AKE}$.
- If Node is corrupted, then Λ can easily find out the random value chosen from Z_q^* for calculating the private and public keys, and hence the value of the private keys are compromised.

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

6 Experimental Setup and Results

In this section, we describe an experimental evaluation of the effectiveness of the protocol on an IoT testbed, including the incurred hardware and performance overheads.

6.1 Attack Scenario and Experimental Setup

We consider a scenario whereby a video camera transmits unencrypted captured video over a network. An adversary intercepts the network traffic to launch "man-in-the-middle attack" and "replay attack", to potentially modify the information received at the receiver. To prevent this, the camera in conjunction with an embedded board and a PUF mapped on a FPGA *emulates* as an IoT node. The scenario is illustrated in Fig. 6. The off-the-shelf hardware components used in the setup are: an *Intel Edison* embedded development platform, a *Digilent Nexys-4* FPGA board containing *Xilinx Artix-7* FPGA, and a *Logitech* HD UVC camera as shown in Fig. 7.

Fig. 7. Experimental setup for smart IoT node.

In general scenario, the Logitech camera is connected to Intel Edison Board through a USB interface to form an IoT node. An mjpg-streamer software is run on the Edison board to capture video using the camera, and send to a PC (the receiver) through WiFi. The PC then displays the received video in a web browser using the IP address of the Edison board. Next, we use the hacking software tools enabled by *Kali Linux*and perform the following steps:

- First, the attacker finds out the IP address of the Edison board from the network ARP table using the *arp* command.
- The video packets are then sniffed using *IP forwarding* and *ettercap* tool and saved in the attacker's machine using the *driftnet* tool.
- The attacker starts scanning the network in *monitor mode* to get the router's BSSID and associated clients using the *airmon-ng* and *airodump-ng* tools.
- Next, it de-authenticates the Edison board from the network using the *deauth* option in the *aireplay-ng* tool. Once this is done, the video stream stops at PC's end for a short interval of time.
- Then, the attacker spoofs the IP address of the Edison board and starts streaming the pre-captured video using the same mjpg-streamer tool.
- Now, the receiver PC actually gets data from the attacker's computer, which can either be a replayed or modified version of the video stream captured earlier.

To prevent this, we have adapted the idea of Double Arbiter PUF [32], designed a 5-4 DAPUF as shown in Fig. 8 and implemented it on Xilinx Artix- 7 FPGAs. The 5-4 DAPUF comprises of five 64 bit Arbiter PUF instances. Each APUF instance consists of two identical delay paths, and let us denote the outputs of top and bottom paths as $P_{i,T}$ and $P_{i,B}$, respectively, where $i = 1, \ldots, 5$. For $i \in \{1, \ldots, 5\}$ and $j \in \{i+1, \ldots, 5\}$, an

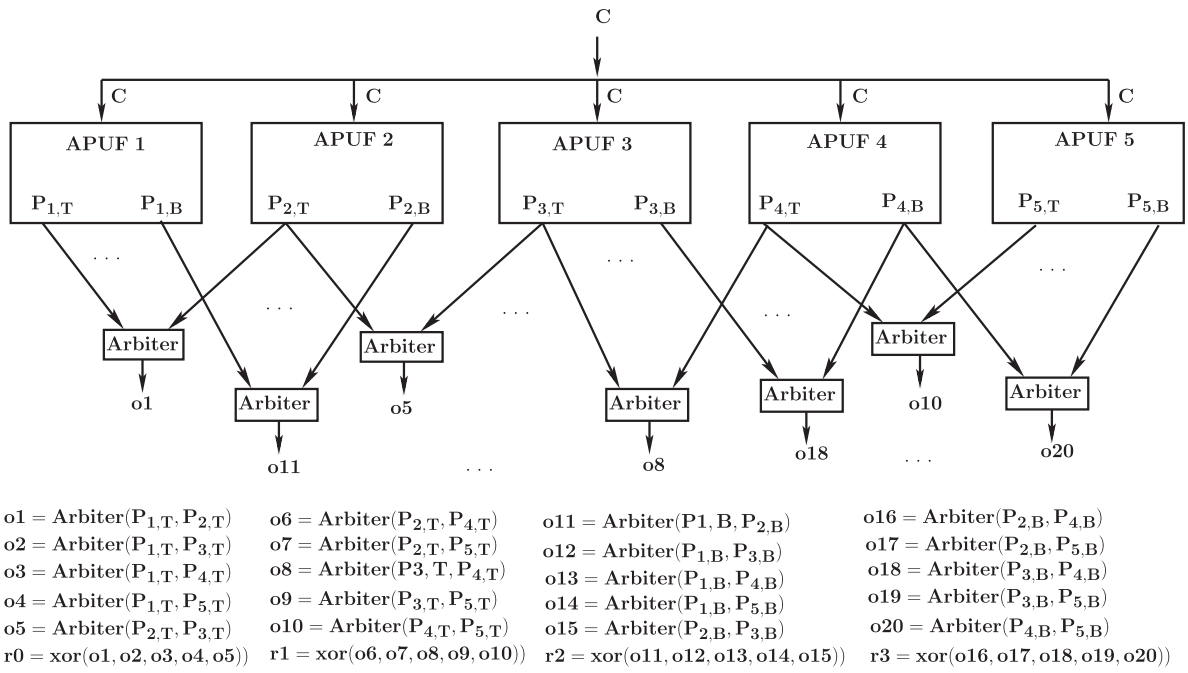

Fig. 8. Architectural overview of 5-4 DAPUF. It generates 4-bit output (r0, r1, r2, r3) to a given challenge, and r_i depends on the outputs of five consecutive arbiters.

arbiter $\text{Arbiter}(P_{i,T}, P_{j,T})$ is instantiated, where the inputs to the arbiter are top paths of ith and jth APUFs. The process is repeated for the bottom paths. Hence, in total 20 arbiters are used in the design. Four 5-input XOR gates are used to generate 4-bit output from the outputs of 20 arbiters, to a given challenge. The Edison board, Artix-7 FPGA and the camera together form a smart IoT node and can act as a prover. The receiver PC acts as the verifier that can generate and validate the response of the PUF instance, and subsequently authenticate the IoT node. Now, with the modified set up, the system works as follows: before streaming the video in the web page, the PC first authenticates the Edison board using our proposed protocol and validates the public keys. Later, if the attacker de-authenticates the Edison board from the network, the video streaming will stop at PC's end. Before reloading the web page, the PC again re-authenticates the device of the video source. This is where the adversary fails to authenticate herself as she does not possess the correct PUF instance.

6.2 Experimental Results

The PUF, BCH encoder and decoder design and implementation was performed using *Xilinx ISE* (v 14.2) design environment. The power consumption of the circuit reported by *Xilinx XPower Analyser* CAD software tool was 0.044 W. We have tested the PUF circuit using CME Nano-Bench Top Chamber (Sl. No. 120433) where 10,000 CRPs of 8 PUF instances are collected 15 times for the temperature variation from -20 to 80°C keeping the other reliability influencing factors such as supply voltage unchanged. Fig. 9 shows the reliability variation of 5-4 DAPUF across the temperatures after error correction, approximately from 97 to 99 percent. One strategy that can be taken to distinguish between a false negative and a true negative is: i) If the authentication passes then it is correct largely. ii) If the authentication fails, there is a chance that it is a false negative. In that case, the verifier can repeat for n times. Let us assume that on average the reliability is 98 percent. Then probability of false negative for one protocol run is = $[1 - 0.98]$ = 0.02. If the verifier repeats the protocol run for 3 times, then the probability of false negative= $(0.02)^3$= 0.00008, which is almost zero. Next, the uniqueness of the deployed 5-4 DAPUF is reported as 44.16 percent. The modelling accuracy of the entire 4-bit response of the PUF is approximately 39 percent using $\mathbf{2 \times 10^5}$ raw CRPs. Finally Table 1 provides a comparative study of hardware and performance overhead of previously discussed PUF-based authentication protocols with our scheme.For software implementation, we used the *MIRACL Crypto SDK*, which provides a C++ software library for elliptic curve cryptography. The specification of the *Cocks-Pinch* curve which has been used for Type 2 Tate pairing is as follows:

- The curve is non-supersingular.
- 512 bit prime number p=8D5006492B424C09D2FE-BE717EE382A57EBE 3A352FC383E1AC79F21DDB43 706CFB192333A7E9CF644636332E83D90A1E56EFBA E8715AA07883483F8267E80ED3
- The equation of the curve is: $y^2 = x^3 + Ax + B$ where: A=-3 and B=609993837367998001C95B87A6-BA872135E26906DB4C192D6E038486177A3EDF6C5

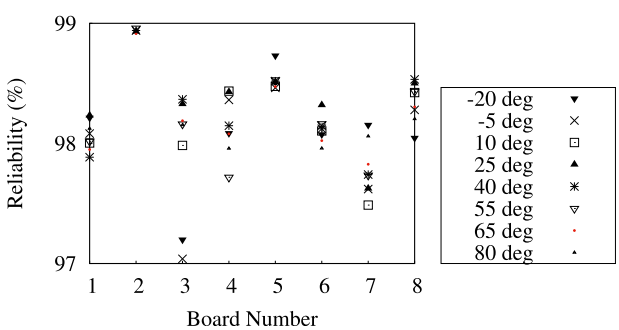

Fig. 9. Reliability of DAPUF across temperature variations.

TABLE 1
Hardware, Storage and Execution Time Overhead Comparison with State-of-the-Art PUF Based Protocols

Protocols	Hardware Overhead	Error Correction Reported	Storage per CRP entry	end-to-end execution time
[8]	4454 Slices on FPGA	not reported	63 bits	not reported
[9]	652 LUTs and 278 registers in FPGA	not reported	63 bits	not reported
[12]	9207 LUTs and 2921 registers in FPGA	not reported	68 bits	not reported
[14]	2061 GE in ASIC	reported	192 bits	not reported
[18]	6034 LUTs and 1724 FFs in FPGA	not reported	2112 bits	2.75 secs
[15]	1037 LUTs and 627 registers in FPGA	not reported	129 bits	not reported
[19]	64kB in SRAM	not reported	984 bits	not reported
[31]	807 slices, 124 registers and 995 LUTs in FPGA	not reported	112 bits	not reported
Proposed work	**PUF:** 456 slices, 283 registers and 887 LUTs; **BCH Encoder:** 41 slices, 35 registers and 19 LUTs; **BCH Decoder:** 1236 slices, 1615 registers and 685 LUTs in FPGA	reported	512 bits	480.11 ms

0B9BB20DF881F2BD05842F598F3E037B362DBF89F0A62E5871D41D951BF8E

- Order q=800000000000000000000000000000000000200001
- The pairing uses an embedding degree of k=2, so the pairing e(P,Q) evaluates naturally as an element in F_{p^2}. P is a point on the elliptic curve $E(F_p)$ and Q is a point on $E'(F_{p^{k/2}})$, or in this case $E'(F_p)$ where E' is the twisted curve. Using compression the pairing evaluates as an element in $F_{p^{k/2}}$, or just F_p in this case.

We ported our implementations to the Edison platform. Overall, the executable took approximately 512 kB of memory on the Edison board. The latency overhead incurred running the end-to-end authentication scheme before the video streaming was 480.11 ms on average. These overhead results demonstrates that the proposed protocol can be implemented while incurring acceptable resource and performance overheads.

7 Conclusions

We have developed a secure PUF based authentication and certificate-less identity based key exchange protocol. Formal security proofs for the protocol have been formulated under the SK security and UC framework. The asymmetric nature of the protocol overcomes the shortcomings of previously proposed CRP based PUF authentication mechanism and suits appropriately in a distributed IoT framework. We have also demonstrated an attack on a prototype video surveillance system, and showed how the proposed scheme can be useful in mitigating the security vulnerability at low hardware and performance overheads. In future, our research work will be directed towards optimization of the resources for frame encryption and investigating side-channel attacks on the proposed protocol implementation.

Acknowledgments

This work was supported partially by Wipro Limited, Information security education Awareness Program (ISEA), DIT, India. Debdeep Mukhopadhyay would like to thank DST Swarnajayanti Fellowship.

References

[1] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.
[2] D. Mukhopadhyay, "PUFs as promising tools for security in internet of things," *IEEE Des. Test*, vol. 33, no. 3, pp. 103–115, Jun. 2016.
[3] D. Lim, J. W. Lee, B. Gassend, G. Edward Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Trans. VLSI Syst.*, vol. 13, no. 10, pp. 1200–1205, 2005.
[4] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. 42nd Annu. Symp. Found. Comput. Sci.*, 2001, pp. 136–145.
[5] J. Delvaux, D. Gu, I. Verbauwhede, M. Hiller, and M. M. Yu, "Efficient fuzzy extraction of PUF-induced secrets: Theory and applications," in *Proc. 18th Int. Conf. Cryptographic Hardware Embedded Syst.*, 2016, pp. 412–431.
[6] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas, "Controlled physical random functions," in *Proc. 18th Annu. Comput. Secur. Appl. Conf.*, 2002, pp. 149–160.
[7] E. Öztürk, G. Hammouri, and B. Sunar, "Towards robust low cost authentication for pervasive devices," in *Proc. 6th Annu. IEEE Int. Conf. Pervasive Comput. Commun.*, 2008, pp. 170–178.
[8] S. Katzenbeisser, Ü. Koçabas, V. van der Leest, A. Sadeghi, G. J. Schrijen, and C. Wachsmann, "Recyclable PUFs: Logically reconfigurable PUFs," *J. Cryptographic Eng.*, vol. 1, no. 3, pp. 177–186, 2011.
[9] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching," in *Proc. IEEE Symp. Secur. Privacy Workshops*, 2012, pp. 33–44.
[10] Ü. Koçabas, A. Peter, S. Katzenbeisser, and A. Sadeghi, "Converse PUF-based authentication," in *Proc. 5th Int. Conf. Trust Trustworthy Comput.*, 2012, pp. 142–158.
[11] M. van Dijk and U. Rührmair, "Physical unclonable functions in cryptographic protocols: Security proofs and impossibility results," *IACR Cryptology ePrint Archive*, vol. 2012, 2012, Art. no. 228.
[12] J. Kong, F. Koushanfar, P. K. Pendyala, A. Sadeghi, and C. Wachsmann, "PUFatt: Embedded platform attestation based on novel processor-based PUFs," in *Proc. 51st Annu. Des. Autom. Conf.*, 2014, pp. 109:1–109:6.
[13] S. Schulz, A. Schaller, F. Kohnhäuser, and S. Katzenbeisser, "Boot attestation: Secure remote reporting with off-the-shelf IoT sensors," in *Proc. 22nd Eur. Symp. Res. Comput. Secur. Comput. Secur.*, 2017, pp. 437–455.
[14] Y. Lao, B. Yuan, C. H. Kim, and K. K. Parhi, "Reliable PUF-based local authentication with self-correction," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 2, pp. 201–213, Feb. 2017.
[15] M. Barbareschi, P. Bagnasco, and A. Mazzeo, "Authenticating IoT devices with physically unclonable functions models," in *Proc. 10th Int. Conf. P2P Parallel Grid Cloud Internet Comput.*, 2015, pp. 563–567.
[16] M. Aman, K. C. Chua, and B. Sikdar, "Mutual authentication in IoT systems using physical unclonable functions," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1327–1340, Oct. 2017.

[17] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Secure lightweight entity authentication with strong PUFs: Mission impossible?" in *Proc. 16th Int. Workshop Cryptographic Hardware Embedded Syst.*, 2014, pp. 451–475.
[18] W. Che, M. Martin, G. Pocklassery, V. K. Kajuluri, F. Saqib, and J. Plusquellic, "A privacy-preserving, mutual PUF-based authentication protocol," *Cryptography*, vol. 1, no. 1, pp. 1–17, 2016, Art. no. 3.
[19] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "Physical unclonable functions, FPGAs and public-key Crypto for IP protection," in *Proc. Int. Conf. Field Programmable Logic Appl.*, 2007, pp. 189–195.
[20] J. R. Wallrabenstein, "Practical and secure IoT device authentication using physical unclonable functions," in *Proc. IEEE 4th Int. Conf. Future Internet Things Cloud*, 2016, pp. 99–106.
[21] A. Costin, "Security of CCTV and video surveillance systems: Threats, vulnerabilities, attacks, and mitigations," in *Proc. 6th Int. Workshop Trustworthy Embedded Devices*, 2016, pp. 45–54.
[22] H. Li, Y. He, L. Sun, X. Cheng, and J. Yu, "Side-channel information leakage of encrypted video stream in video surveillance systems," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[23] U. L. Puvvadi, K. Di Benedetto, A. Patil, K.-D. Kang, and Y. Park, "Cost-effective security support in real-time video surveillance," *IEEE Trans. Ind. Informat.*, vol. 11, no. 6, pp. 1457–1465, Dec. 2015.
[24] T.-S. Park and M.-S. Jun, "User authentication protocol for blocking malicious user in network CCTV environment," in *Proc. 6th Int. Conf. Comput. Sci. Convergence Inf. Technol.*, 2011, pp. 18–24.
[25] U. Chatterjee, R. S. Chakraborty, H. Kapoor, and D. Mukhopadhyay, "Theory and application of delay constraints in arbiter PUF," *ACM Trans. Embedded Comput. Syst.*, vol. 15, no. 1, pp. 10:1–10:20, 2016.
[26] Infenion, "Trusted platform module fundamental," 2008. [Online]. Available: http://cs.unh.edu/ it666/reading_list/Hardware/tpm_fundamentals.pdf
[27] N. Asokan, F. F. Brasser, A. Ibrahim, A. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, "SEDA: Scalable embedded device attestation," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 964–975.
[28] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning physically unclonable functions," in *Proc. IEEE Int. Symp. Hardware-Oriented Secur. Trust*, 2013, pp. 1–6.
[29] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, 2010, pp. 237–249.
[30] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. London, U.K./Boca Raton, FL, USA: Chapman and Hall/CRC Press, 2007.
[31] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-based secure communication protocol for IoT," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 3, pp. 67:1–67:25, 2017.
[32] Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede, "Elliptic-curve-based security processor for RFID," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1514–1527, Nov. 2008.

Urbi Chatterjee is working toward the PhD degree in the Indian Institute of Technology Kharagpur, India, since 2015. Before that, she worked as assistant systems engineer with TATA Consultancy Services Limited, Kolkata. Her research interests include design of PUF based lightweight authentication and secure communication protocols, cryptanalysis, and security evaluation of PUFs.

Vidya Govindan is working toward the master's degree in the Computer Science and Engineering Department, Indian Institute of Technology Kharagpur, India. Prior to that she had worked as hardware design engineer with Tonbo Imaging Pvt Ltd, Bangalore, India. Her current research focuses on security of IOT and embedded systems.

Rajat Sadhukhan is working toward the PhD degree in the Indian Institute of Technology-Kharagpur, India, since 2016. Prior to joining research programme he has worked with Intel Technology India Pvt. Ltd., Bangalore for seven years. His research interests include symmetric key cryptography, hardware security, and VLSI design.

Debdeep Mukhopadhyay received the PhD degree from the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India, in 2007, where he is currently an associate professor. His research interests include cryptography, VLSI of cryptographic algorithms, hardware security, and side channel analysis.

Rajat Subhra Chakraborty is an associate professor with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. His area of research is hardware security, VLSI design (especially low-power and robust design) and digital content protection through watermarking. He is a senior member of the IEEE and ACM.

Debashis Mahata received the MSc degree in physics from Burdwan University, West Bengal and the MTech degree in computer science from the Indian Statistical Institute, Kolkata. He is a distinguished member of Technical staff-senior member, with Wipro Technologies. His current areas of interests include connected devices security, neural networks, and video collaboration.

Mukesh M. Prabhu received the MS degree from IIT Madras. He is a distinguished member of Technical staff and head of the IP & Innovation of Product Engineering Services, Wipro Technologies. His current areas of interests include connected devices security, augmented reality, video collaboration, designing end-to end systems, and applications addressing business challenges.