PUF+IBE: Blending Physically Unclonable Functions with Identity Based Encryption for Authentication and Key Exchange in IoTs

Urbi Chatterjee, Vidya Govindan, Rajat Sadhukhan, Debdeep Mukhopadhyay, *Member, IEEE,* Rajat Subhra Chakraborty, *Senior Member, IEEE,* Debashis Mahata, and Mukesh 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, its unclonability property puts forward an open problem of how to develop such mechanism without needing to store the challenge-response pair (CRP) explicitly at the verifier end.

In this paper, we develop an authentication and key exchange protocol called *PUF+IBE* by combining the ideas of Identity based Encryption (IBE) using Cryptographic Pairing with PUFs. We show that this combination can help to do away with the requirement of explicitly storing the secret CRPs. The proposed protocol is based on two security assumptions: (a) physical and mathematical unclonability of the constituent PUFs, and, (b) computational intractability of the *Elliptic Curve Discrete Logarithm Problem* (ECDLP). The security of the protocol is proved in a formal way 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 comprising 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 as presented in [1], 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 PUF and IBE [2]. PUFs can be proposed as an unconventional, lightweight hardware security primitive that provides reasonable degree of security [3], [4], [5]. A Silicon PUF [6], [7] 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 of the device. PUFs have been proposed in various security applications such as IC anti-counterfeiting, device identification and authentication [8], binding hardware to software platforms [9], secure storage of cryptographic secrets [10], keyless secure communication [11], etc. 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 a "challenge-response authentication" mechanism, rather than on a single secret cryptographic key. We use this property to uniquely identify each devices in the IoT framework. The identity generated by the PUF is further used to generate private/public key pairs for certificate less identity based encrytion; thus offloads the complexity of managing the keys for each of them. In summary, we make the following contributions:

- We propose an authentication and key exchange protocol combining the concepts of PUF and IBE, called *PUF+IBE*. The protocol solves an open problem in the

- *Urbi Chatterjee, Vidya Govindan, Rajat Sadhukhan, Debdeep Mukhopadhyay and Rajat Subhra Chakraborty are the members of Secure Embedded Architecture Laboratory (SEAL), Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur,Kharagpur,India-721302.*
  E-mail: urbi.chatterjee@cse.iitkgp.ernet.in, {vidya.govindan, rajat.sadhukhan}@iitkgp.ac.in, {debdeep, rschakraborty}@cse.iitkgp.ernet.in
- *Debashis Mahata and Mukesh Prabhu are the members of Wipro Technologies, India.*
  E-mail:{debashis.mahata, mukesh.prabhu}@wipro.com

domain of PUF based protocols, alleviating necessity of the verifier to store explicitly the CRP of the PUF. Because of the properties of PUF, the prover nodes do not need to explicitly store keys, but rather generate it at run time, based on which its public and private key are generated. Due to the properties of IBE, these keys are verified by the verifier, without requiring explicit certificate management. The protocol is also heterogeneous, in the sense that the verifier, which is often a server, does not require possession of a PUF and operates with a standard keyed hash function.

- We prove formally the security of the protocol in the Session Key Security model and the Universal Composability framework [12], [13].
- We implement a prototype of the protocol to securely authenticate a video surveillance camera, commercially purchased and devoid of any inbuilt security feature. We make a hardware-software co-deisgn of the prototype, by connecting the camera to a 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. 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 Section 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 server and communicate with the server 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 server, is assumed to have the capability to compute keyed hash function, where the key is stored in a Non-volatile memory.

Protocol 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 server on behalf of the legitimate nodes, without possession of the node. The PUF challenge-response, which is embedded in the node is not accessible to the adversary. Further, the attacker can corrupt the server (as by a malware) and can obtain access to the databases which the server possesses. However, we assume that the attacker cannot gain knowledge of the secret key. We consider man-in-the-middle attacks on the authentication protocol. The attack is successful if the adversary is able to authenticate a wrong node by communicating messages based on the eavesdropped messages sent by legitimate nodes, or by corrupting the server. The protocol assumes that there is an initial enrolment phase when the node is registered in a trusted environment, but when the protocol commences between the nodes and the verifying server, the databases are accessible to the adversary. This makes the model more practical for IoTs as in many such realistic scenarios the server can be corrupted by adversaries and the server may not have a PUF hardware. 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 [2].

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':** A primary objective of the protocol is that instead of having explicit key storage, a PUF instance will be embedded in each IoT data nodes which will act in an unpredictable but repetitive way to provide unique identity to the devices.
- **Lightweightedness and minimal overhead on execution time :** The hardware overhead and power-consumption of the PUF enabled node and the latency to authenticate a legitimate node should be very less.
- **No explicit storage of CRP with verifier:** Unlike traditional PUF-based authentication protocols the verifier will not have access to the 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.
- **Efficient management of public/ private keys without central authority:** The scheme should be designed in such a way that 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 [14], [15], [16], [17], [18], [19] have been proposed in the recent past. But in [20], the authors demonstrated several vulnerabilities such as Denial-of-Service (DoS) attack, synchronization problem, replay attack, token/server impersonation that have made these protocols unacceptable

in their original form. 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 set-up in a hierarchical network of IoT framework, it may expand the attack surface substantially, as the integrity of CRP details at lower level network 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 through an identity based authentication and key exchange protocol, utilizing the device-specific behaviour of PUFs. 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 which is often the data collecting node, does not need to explicitly store any key, but 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 authentication is by the use of Public Key Infrastructure (PKI) or by the use of Identity Based Encryptions (IBE). In [21], 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 of 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 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. Current availability of several lightweight hardware and software implementations [22], [23], [24], [25], [26], [27], [28], [29], [30], for elliptic curve cryptography and bilinear pairing, and lightweight designs for PUFs make such a protocol require less hardware area, consume low power and yet have low latency.

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 [31], [32], [33]. Till now, several passive and active attacks [34], [35] 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 [36], [37] were proposed in 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 [38], [39]. 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 [40]. Moreover, in [41], Zheng et. al. shows how Dynamic Vision Sensor based PUF can be used for identification and secret key generation for cameras used in reactive monitoring system. 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 PUF+IBE: 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 PUF IoT Node. Smart IoT nodes, which play the role of *prover*, reside at the lowest level of the architecture. In our proposal, we assume these data 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*. These verifier nodes are relatively resourceful, and hence can execute both classical and unconventional security protocols. As shown in Fig. 1, we assume that the network infrastructure supports PUF based authentication and key exchange protocol between the prover data nodes at the leaf level and the verifier nodes. On the other hand, the verifier and all the upper layer nodes follow traditional security algorithms for data confidentiality and integrity.

The proposed protocol has two main phases: In the **Enrolment phase**, a CRP database is generated for each of the IoT data nodes (marked as '1'). The CRP databases (CRPDBs) are assumed to be stored in a secure database in a trusted

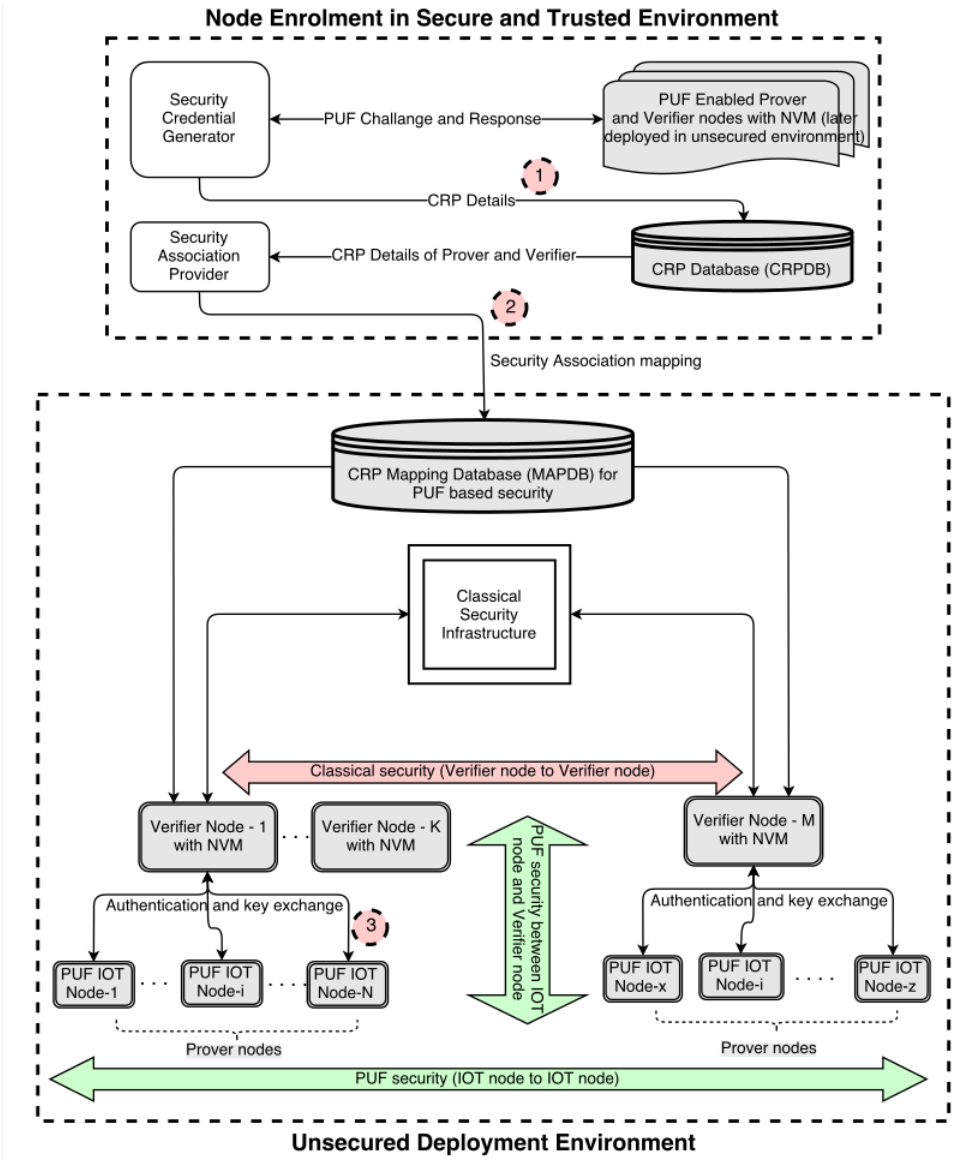

Fig. 1. Hierarchical IoT architecture and proposed secure communication mechanism.

environment, outside the reach of the typical IoT communications. These database entries are never directly used for authentication. Rather a security relationship mapping is created (marked as '2') for each CRP which hides the input-output correlation of the PUF instance. These mapping entries are stored in CRP Mapping Databases (MAPDBs) maintained outside the trusted environment. Additionally, a randomly generated key is provided to the verifier for each prover node. It is stored in a secure NVM and used to dynamically generate a mapped response for a particular challenge of a PUF instance. In the **Authentication and key exchange phase**, the verifier uses challenges randomly selected from MAPDB 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 between different prover nodes for generation and sharing of public keys (marked as '3').

Fig. 2 demonstrates the situation when two prover nodes try to communicate with each other, their corresponding verifier nodes first authenticate them. Then, public keys are exchanged between the verifier nodes, followed by information exchange between the prover nodes *via the verifier nodes*. The secrecy and integrity of the messages are ensured by the classical security mechanisms. In case of mobile data nodes, if a node moves to a new location, then it will be connected to a new verifier. The request/response between the actual verifier and the prover would be routed using the classical security enabled communication channel.

We now describe details of the steps and constructs used in the proposed security protocol.

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 pair-

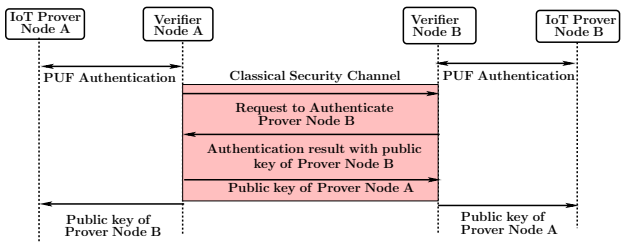

Fig. 2. Intergroup communication in the proposed protocol.

ing. Pairing is an admissible bilinear map ê: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ which satisfies the following three properties:

1) **Bilinearity**: $\forall a, b \in F_q^*, \forall P \in \mathbb{G}_1, Q \in \mathbb{G}_1 : e(aP, bQ) = e(P, Q)^{ab}$.
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 [42]. 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: $<p$, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_3$, 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. 3, and are summarized as follows:

- A *Security Credential Generator* (SCG) 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 stores the response along with the challenge by appending $<C_A, R_A>$ to its CRPDB.
- Next, the SCG uniquely selects an m-bit key K_A for node A and randomly generates an n-bit challenge C_S, and then it calculates:

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

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

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

  In this way, a new tuple $< C_a, C_S, a, B, d_1 >$ is generated and stored in the MAPDB of the *Security Association Provider* (SAP). This procedure is repeated according to the memory capacity of the SAP and the SCG.
- Finally, the key K_A along with the index of Node A are stored in the secure non-volatile memory of the verifier.

At the end of the enrolment phase for a given node A, the verifier 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. The verifier will calculate the response of the PUF on-the-fly to authenticate node A. Here, we have assumed that the verifier 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) [43]

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. 4. 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 IoT hierarchy.

- At first, IoT node A initiates a request to the verifier for authentication. The verifier forwards the request to the SAP.
- The SAP randomly chooses an entry $< C_A, C_S, a, B, d_1 >$ from $MapDB_A$ and sends it back to the verifier.
- Now, the verifier performs the following computations:

  $$P_S = H2_{K_A}(C_S)$$

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

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

- Next, the verifier randomly chooses a value x such that $x \in_R Z_q^*$ and computes:

  $$Q_A = P_A + x \cdot P_S, V_A = e(P_A, x \cdot P_S)$$

  and sends this value to node A as the tuple $< C_A, Q_A >$.
- On receiving the message, node A first calculates the following:

  $$P'_A = H1(PUF_A(C_A)), P'_S = Q_A - P'_A, V'_A = e(P'_A, P'_S)$$

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

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

  and it sends the verifier the tuple $< V'_A, KA_{PUB}, R_A, H3(P'_S + KA_{PUB} + R_A) >$.
- If V_A equals V'_A and $H3(P'_S + KA_{PUB} + R_A)$ equals $H3(x \cdot P_S + KA_{PUB} + R_A)$, the verifier 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 to authenticate node B. Hence, the verifier 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 $< KB_{PUB}, Q_B, R_B, H3(P_A + KB_{PUB} + Q_B + R_B) >$. On receiving it, if node A finds that $H3(P_A + KB_{PUB} + Q_B + R_B)$ equals $H3(P'_A + KB_{PUB} + Q_B + R_B)$, then the verifier is authenticated, as only the verifier can retrieve the value of P_A using P_S, and node A accepts the public key of node B.

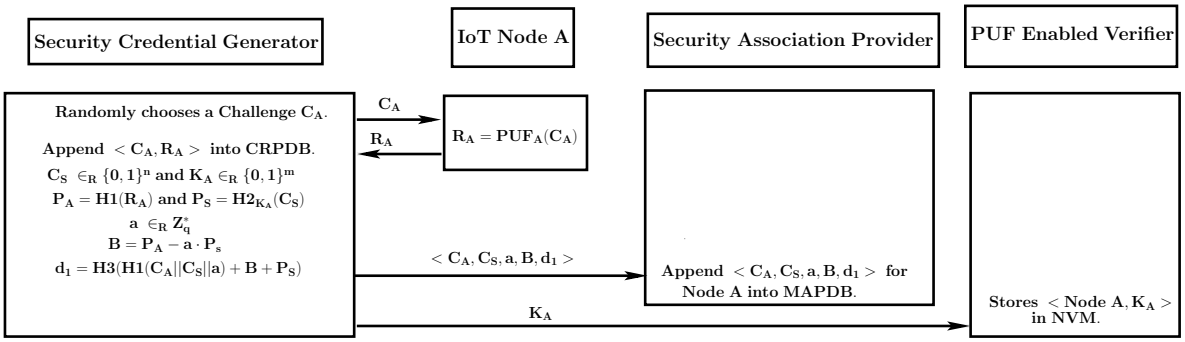

Fig. 3. Enrolment phase of the proposed protocol.

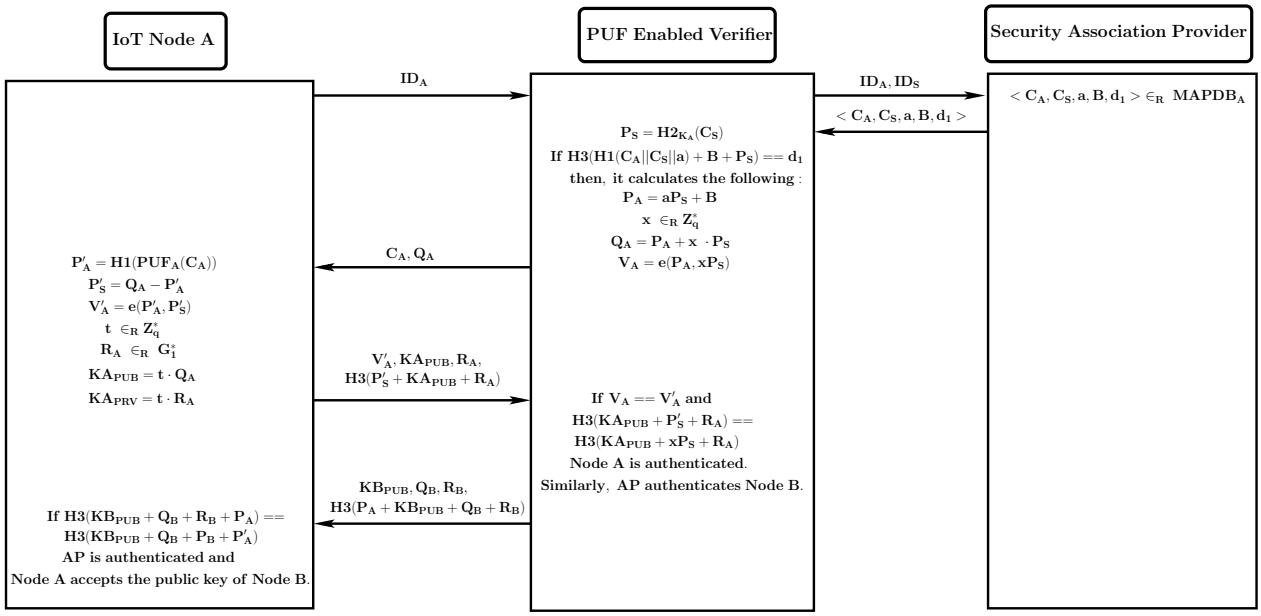

Fig. 4. The authentication and key exchange phase.

5 Security Analysis

Attack Models: Next we will shortly describe two different attack models in which we will analyse the security of the proposed authentication and key exchange protocol.

- ***Session Key Security Model*:** Here all parties involved in the protocol such as SCG, SAP, verifier and PUF IoT Nodes 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. The primary concepts of this framework are described in Section 5.2. We have described an ideal functionality of the asymmetric key exchange protocol in Section 5.2.1. 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 different scenarios where

1) The verifier and the two communicating parties are honest (ideal case).
2) The verifier 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).

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, chosen from the probability distribution of keys generated by the protocol π. 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 arbitary 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 [44]), successful mathematical modeling ("model-building attacks") have been widely reported [45]. However, by choosing relatively secure PUF variants such as *Lightweight Secure PUF* or *XOR PUF* [45], 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:

$$\begin{aligned} &\mid Pr[\mathcal{A}(C, PUF_{Adv}, z) = 1] - \\ &Pr[\mathcal{A}(C, PUF_{Adv}, PUF_N(C)) = 1] \mid \leqslant negl(n) \end{aligned}$$

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 (ECDLP))** 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 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_A. Moreover, let $output_{nodeA,\pi}(C_A, Q_A)$ and $output_{S,\pi}(C_A, C_S, a, B)$ denote the respective outputs of node A and the verifier. 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,

$$\begin{aligned} output_{nodeA,\pi}(C_A, Q_A) = e(&H1(PUF_A(C_A), \\ &Q_A - H1(PUF_A(C_A)) \end{aligned}$$

and

$$output_{S,\pi}(C_A, C_S, a, B) = e(a \cdot H2_{K_A}(C_S) + B, x \cdot H2_{K_A}(C_S))$$

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

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:

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

It can be observed that:

$$e(a \cdot H2_{K_A}(C_S) + B, x \cdot H2_{K_A}(C_S)) = e(P_A, x \cdot H2_{K_A}(C_S)) = e((H1(PUF_A(C_A)), Q_A - H1(PUF_A(C_A))$$

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 [2]. 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 $\lambda = e(KB_{PUB}, R_B)^w = e(t \cdot Q_B, R_B)^w$=$e(Q_B, R_B)^{t \cdot w}$, 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 $e(w \cdot Q_B, KB_{PRV})$=$e(w \cdot Q_B, t \cdot R_B)$=$e(Q_B, R_B)^{t \cdot w}$=$\lambda$. 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 $< C_A, C_S, a, B >$ and x chosen randomly. This has been formulated by giving an adversary the values $< C_A, C_S, a, B >$ 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 ζ=$< C_A, C_S, a, B, Q_A >$ where $Q_A = ((a \cdot H2_{K_A}(C_S) + B) + (x \cdot H2_{K_A}(C_S)))$.
   b) Two identities V_{A_0} and V_{A_1}, calculated based on a chosen random bit $b \in \{0, 1\}$:

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

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 1. 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:

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

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 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:

$$Q_A = P_A + x \cdot P_S$$

   d) Then sets $\zeta = < C_A, C_S, a, B, Q_A >$, 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 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. Hence,

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

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 $K1_{PRV} = t \cdot R_A$. 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.

$\implies$ π is not SK-secure $\rightarrow \neg$ DUP $\vee \neg$ ECDLP.

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 [13]:

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.
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. 6. Next we prove the security of $\mathcal{F}_{AKE}$.

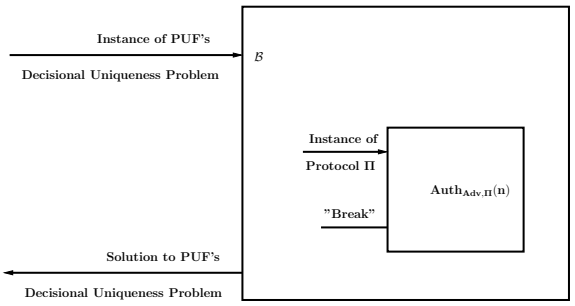

Fig. 5. The view of $\mathbf{Auth_{adv},\pi}$ when it is run as a sub-routine of $\mathcal{B}$ ([46]).

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

***proof* 2.**

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 ($\mathbf{establish-session_{AKE}}$, $\mathbf{sid}$, $\mathbf{Node\ A}$, $\mathbf{Verifier}$) for the first time, the simulator $\mathcal{S}$ queries the PUF instance PUF_A for k random challenges C_1, C_2,..., C_k, and obtains responses R_{A1}, R_{A2},..., 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 ($\mathbf{establish-session_{AKE}}$, $\mathbf{sid}$, $\mathbf{Node\ A}$, $\mathbf{Verifier}$), $\mathcal{F}_{AKE}$ 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 server sends (C_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 $\mathcal{F}_{AKE}$.
- The probability that the value of $e((PUF_A(C_A), x \cdot H2_{K_A}(C_S))$ 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 (C_A, Q_A), then the simulator $\mathcal{S}$ can easily evaluate

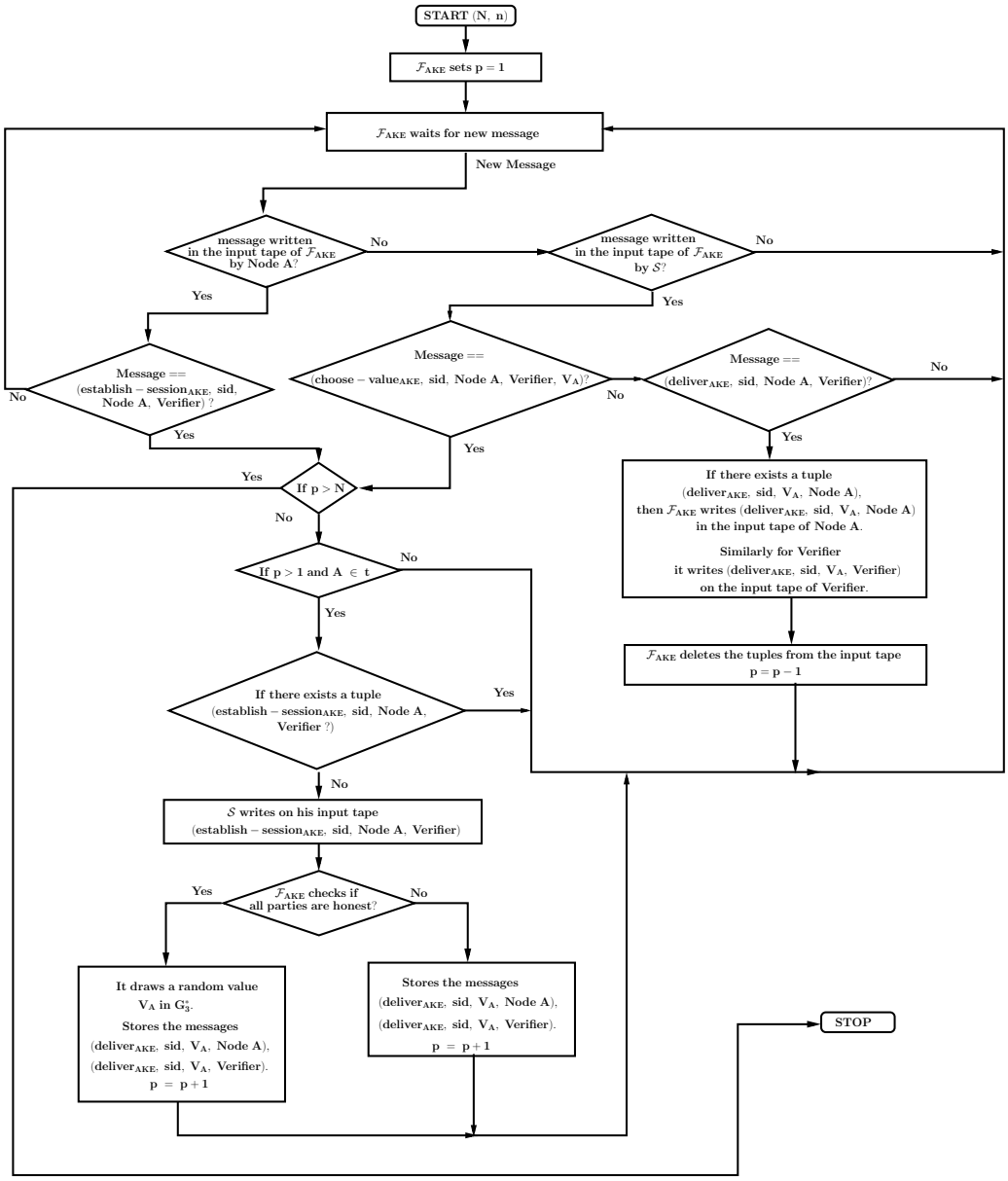

Fig. 6. The asymmetric key exchange ideal functionality.

$V_A = e(a \cdot H2_{K_A}(C_S) + B, x \cdot H2_{K_A}(C_S))$, as the server is corrupt. But it is to be noted that $\mathcal{S}$ does not have access to K_A. It can only get the final value of V_A.

- 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, R_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.

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 server sending (C_A, V_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 proposed protocol to prevent an attack on an IoT application, including the incurred hardware and performance overheads.

6.1 Attack Scenario

We consider a scenario whereby a video camera transmits unencrypted captured video over a network. An adversary intercepts the network traffic to launch a "man-in-the-middle attack" and a "replay attack", to potentially modify the information received at the receiver. To prevent this IP spoofing and impersonation attack, the camera in conjunction with a PUF mapped on a FPGA and an embedded board emulates an IoT node. The scenario is illustrated in Fig. 7.

6.2 Experimental Setup

The off-the-shelf hardware components used in the attack 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. 8.

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. Both the camera and the PC are connected to the same WiFi network. Next, we use the hacking software tools enabled by *Kali Linux* [47] as mentioned below and perform a man-in-the-middle and replay attack on this setup, through 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 these frames are saved in the attacker's machine using the *driftnet* tool as shown in Fig.10.
- 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 as shown in Fig. 10.
- 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.

Next, the PUF circuitry is implemented on the Xilinx Artix-7 FPGA and is connected to the Edison Board. The communication between FPGA and Edison board is established through a *Universal Asynchronous Receive/Transmit* (UART) interface. 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 embedded in Artix-7, 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

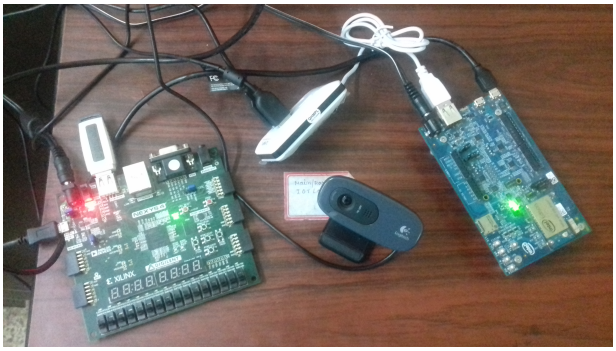

Fig. 8. Experimental setup for smart IoT node.

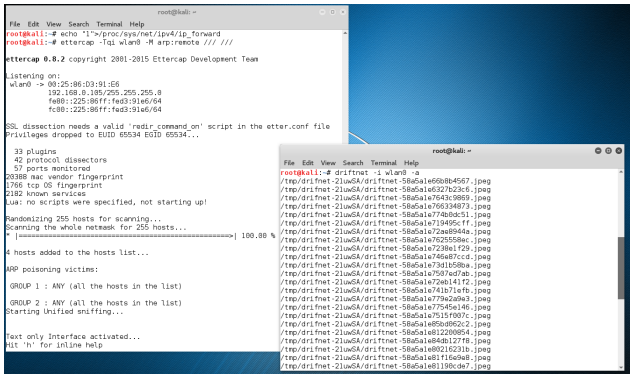

Fig. 9. The video frames are saved in jpeg format in the adversary's machine using driftnet tool which are used for replay.

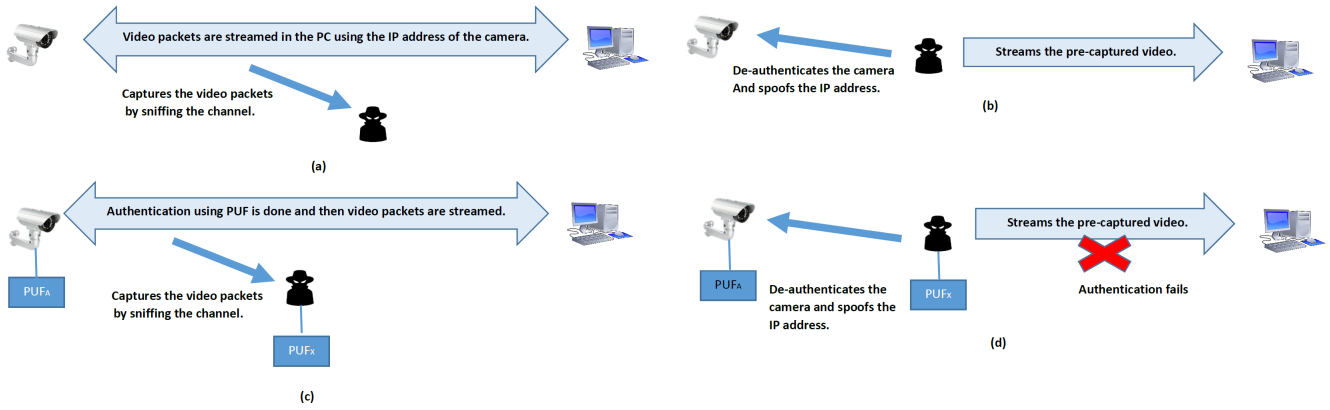

Fig. 7. Attack on video surveillance system and protection against it: (a) and (b) show the successful attack in the absence of PUF based authentication mechanism, while (c) and (d) show the prevention of the attack in the presence of the proposed PUF based authentication system.

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.

6.3 Experimental Results

The PUF design and implementation was performed using *Xilinx ISE* (v 14.2) design environment. We applied a 64-bit challenge as a seed to a circular left shift module. After 10 cycles, the output of the module was input to the LSPUF [48], and the 10-bit output from the PUF was stored. This process was repeated 5 times to get a 50-bit response from the PUF. As hardware overhead for deploying the PUF in the FPGA, the number of required Slices, Registers and LUTs are 547, 288 and 1347 respectively. The power consumption of the circuit reported by *Xilinx XPower Analyzer* CAD software tool was 0.044 W.

For software implementation of the elliptic curve cryptographic operations, we used the *MIRACL Crypto SDK* [49], which provides a C++ software library for elliptic curve cryptography. The specification of the elliptic curve is as follows: the 512 bit prime number p= 8D5006492B424C09D2FEBE717EE382A57EBE3A352FC383E1AC79F21DDB43706CFB192333A7E9CF644636332E83D90A1E56EFBAE8715AA07883483F8267E80ED3, order r=8000000000000000000000000000000020001; the curve being $y^2 = x^3 + Ax + B$ where: A=-3 and B=609993837367998001C95B87A6BA872135E26906DB4C192D6E038486177A3EDF6C50B9BB20DF881F2BD05842F598F3E037B362DBF89F0A62E5871D41D951BF8E. 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 approximately 480.11 ms on average. These overhead results demonstrates that the proposed protocol can be implemented while incurring acceptable resource and performance overhead.

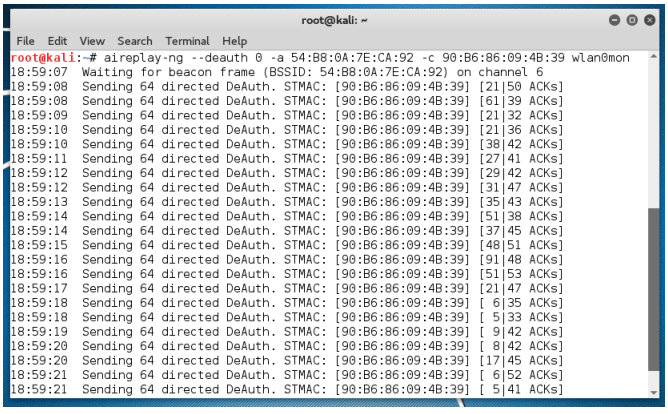

Fig. 10. The camera is de-authenticated from the wireless network by the adversary using aireplay tool (right).

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.

References

[1] A. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," in *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, 2015, pp. 54:1–54:6.

[2] D. Boneh and M. K. Franklin, "Identity-Based Encryption from the Weil Pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[3] NXP, "ME and MY SMARTER WORLD," http://blog.nxp.com/identification/puf-the-magic-dragon-of-smart-card-security.

[4] IoTONE, "PUF The Magic (IoT) Dragon," http://www.iotone.com/guide/puf-the-magic-iot-dragon/g488.

[5] D. Mukhopadhyay, "PUFs as Promising Tools for Security in Internet of Things," *IEEE Design & Test*, vol. 33, no. 3, pp. 103–115, 2016.

[6] R. Pappu, "Physical One-way functions," Ph.D. dissertation, Massachusetts Institute of Technology, USA, 2001.

[7] D. LIM, "Extracting Secret keys from Integrated Circuits," USA, 2004.

[8] M. Majzoobi and F. Koushanfar, "Time-Bounded Authentication of FPGAs," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3-2, pp. 1123–1135, 2011.

[9] S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended Abstract: The Butterfly PUF Protecting IP on every FPGA," in *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust*, ser. HOST '08, 2008, pp. 67–70.

[10] M. M. Yu, D. M'Raïhi, R. Sowell, and S. Devadas, "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*, 2011, pp. 358–373.

[11] U. Rührmair, "SIMPL Systems as a Keyless Cryptographic and Security Primitive," in *Cryptography and Security*. Springer, 2012, pp. 329–354.

[12] R. Canetti and H. Krawczyk, "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*, 2001, pp. 453–474.

[13] R. Canetti, "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*, 2001, pp. 136–145.

[14] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas, "Controlled Physical Random Functions," in *18th Annual Computer Security Applications Conference (ACSAC 2002), 9-13 December 2002, Las Vegas, NV, USA*, 2002, pp. 149–160.

[15] E. Öztürk, G. Hammouri, and B. Sunar, "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*, 2008, pp. 170–178.

[16] S. Katzenbeisser, Ü. Koçabas, V. van der Leest, A. Sadeghi, G. J. Schrijen, and C. Wachsmann, "Recyclable PUFs: logically reconfigurable PUFs," *J. Cryptographic Engineering*, vol. 1, no. 3, pp. 177–186, 2011.

[17] 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 *2012 IEEE Symposium on Security and Privacy Workshops, San Francisco, CA, USA, May 24-25, 2012*, 2012, pp. 33–44.

[18] Ü. Koçabas, A. Peter, S. Katzenbeisser, and A. Sadeghi, "Converse PUF-Based Authentication," in *Trust and Trustworthy Computing - 5th International Conference, 2012, Vienna, Austria, June 13-15, 2012. Proceedings*, 2012, pp. 142–158.

[19] M. van Dijk and U. Rührmair, "Physical unclonable functions in cryptographic protocols: Security proofs and impossibility results," *IACR Cryptology ePrint Archive*, vol. 2012, p. 228, 2012.

[20] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "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*, 2014, pp. 451–475.

[21] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "Physical Unclonable Functions, FPGAs and Public-Key Crypto for IP Protection," in *FPL 2007, International Conference on Field Programmable Logic and Applications, Amsterdam, The Netherlands, 27-29 August 2007*, 2007, pp. 189–195.

[22] Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede, "Elliptic-Curve-Based Security Processor for RFID," *IEEE Trans. Computers*, vol. 57, no. 11, pp. 1514–1527, 2008.

[23] L. Batina, N. Mentens, K. Sakiyama, B. Preneel, and I. Verbauwhede, "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*, 2006, pp. 6–17.

[24] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede, "An Elliptic Curve Processor Suitable For RFID-Tags," *IACR Cryptology ePrint Archive*, vol. 2006, p. 227, 2006.

[25] X. Xiong, D. S. Wong, and X. Deng, "TinyPairing: A Fast and Lightweight Pairing-Based Cryptographic Library for Wireless Sensor Networks," in *2010 IEEE Wireless Communications and Networking Conference, 2010, Proceedings, Sydney, Australia, 18-21 April 2010*, 2010, pp. 1–6.

[26] M. Yoshitomi, T. Takagi, S. Kiyomoto, and T. Tanaka, "Efficient Implementation of the Pairing on Mobilephones Using BREW, journal = IEICE Transactions," vol. 91-D, no. 5, pp. 1330–1337, 2008.

[27] T. Acar, K. E. Lauter, M. Naehrig, and D. Shumow, "Affine Pairings on ARM," *IACR Cryptology ePrint Archive*, vol. 2011, p. 243, 2011.

[28] G. Grewal, R. Azarderakhsh, P. Longa, S. Hu, and D. Jao, "Efficient Implementation of Bilinear Pairings on ARM Processors," in *Selected Areas in Cryptography, 19th International Conference, 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, 2012, pp. 149–165.

[29] M. Shirase, Y. Miyazaki, T. Takagi, D. Han, and D. Choi, "Efficient Implementation of Pairing-Based Cryptography on a Sensor Node," *IEICE Transactions*, vol. 92-D, no. 5, pp. 909–917, 2009.

[30] V. Gupta, M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: A standards-based end-to-end security architecture for the embedded internet," *Pervasive and Mobile Computing*, vol. 1, no. 4, pp. 425 – 445, 2005, special Issue on PerCom 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119205000568

[31] Systemsurveyor, "How the Internet of Things (IoT) Relates to Surveillance and Security," http://systemsurveyor.com/iot_surveillance/.

[32] IFSEC Global, "Smart CCTV and the Internet of Things: 2016 trends and Predictions," https://www.ifsecglobal.com/smart-cctv-and-the-internet-of-things-2016-trends-and-predications/.

[33] CSO, "Report: Surveillance cameras most dangerous IoT devices in enterprise," http://www.csoonline.com/article/3142484/internet-of-things/report-surveillance-cameras-most-dangerous-iot-devices-in-enterprise.html.

[34] A. Costin, "Security of CCTV and Video Surveillance Systems: Threats, Vulnerabilities, Attacks, and Mitigations," in *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices*. ACM, 2016, pp. 45–54.

[35] H. Li, Y. He, L. Sun, X. Cheng, and J. Yu, "Side-channel information leakage of encrypted video stream in video surveillance systems," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[36] U. L. Puvvadi, K. Di Benedetto, A. Patil, K.-D. Kang, and Y. Park, "Cost-effective security support in real-time video surveillance," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 6, pp. 1457–1465, 2015.

[37] T.-S. Park and M.-S. Jun, "User authentication protocol for blocking malicious user in network CCTV environment," in *Computer Sciences and Convergence Information Technology (ICCIT), 2011 6th International Conference on*. IEEE, 2011, pp. 18–24.

[38] Nik Rawlinson, "How to set up a cheap home security system using Dynamic DNS," http://www.cnet.com/how-to/how-to-set-up-a-cheap-home-security-system-using-dynamic-dns/.

[39] Stack Exchange, "Securing remotely accessible IP cameras that do not support HTTPS," http://security.stackexchange.com/questions/56779/securing-remotely-accessible-ip-cameras-that-do-not-support-https.

[40] Wowza Media Systems, "RTP/RTSP over SSL," https://www.wowza.com/forums/showthread.php?34002-RTP-RTSP-over-SSL.

[41] Y. Zheng, Y. Cao, and C. Chang, "A new event-driven dynamic vision sensor based physical unclonable function for camera authentication in reactive monitoring system," in *2016 IEEE Asian Hardware-Oriented Security and Trust, AsianHOST 2016, Yilan, Taiwan, December 19-20, 2016*, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/AsianHOST.2016.7835551

[42] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-based Secure Communication Protocol for IoT," *IACR Cryptology ePrint Archive*, vol. 2016, p. 674, 2016.

[43] Infenion, "Trusted Platform Module Fundamental," http://cs.unh.edu/~it666/reading_list/Hardware/tpm_fundamentals.pdf, 2008.

[44] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning physically unclonable functions," in *Hardware-Oriented Security*

and Trust (HOST), 2013 IEEE International Symposium on. IEEE, 2013, pp. 1–6.

[45] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "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*, 2010, pp. 237–249.

[46] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.

[47] Offensive Security, "Kali Linux: Penatration Testing," http://www.kali.org.

[48] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *2008 International Conference on Computer-Aided Design, ICCAD 2008, San Jose, CA, USA, November 10-13, 2008*, 2008, pp. 670–673.

[49] MIRACL, "MIRACL Crypto Library User Documentation," http://docs.miracl.com/miracl, 2016.