Resource-Efficient and Data Type-Aware Authentication Protocol for Internet of Things Systems

Cong Pu[†] Imtiaz Ahmed[‡] Sumit Chakravarty[¶]
[†]Oklahoma State University, United States. Email: cong.pu@ieee.org
[‡]Howard University, United States. Email: imtiaz.ahmed@howard.edu
[¶]Kennesaw State University, United States. Email: schakra2@kennesaw.edu

***Abstract*—In recent years, Internet of Things (IoT) technology has become an essential part of civil and commercial sectors, which are eager to realize their digital transformation. Generally, IoT devices are deployed to gather various information from their surroundings. Nevertheless, disclosing/compromising IoT devices' data to/by unauthorized entities during transmission might compromise the objectives of IoT applications. In addition, low-cost IoT devices are usually made of small circuit with limited computing power and energy supply, thus, security protocols must not affect IoT devices' and systems' functionalities. Recently, several authentication schemes are developed to enable IoT entities (e.g., devices and gateway) to securely exchange information over insecure wireless channels. However, the existing solutions either incur high computation/communication overhead, or have intrinsic security design flaws, which make IoT systems suffer performance degradation or cyber attacks. Especially, none of them distinguish between the different types of data collected by IoT devices. In this paper, we propose a resource-efficient and data type-aware authentication protocol using Chebyshev polynomials, hereafter referred to as *CHEAP*, for IoT systems. The *CHEAP* consists of two sub-schemes: (i) authentication and key establishment between IoT device and IoT gateway; and (ii) authentication and key establishment between two IoT devices. We verify the security properties of *CHEAP* on AVISPA, and the verification results indicate that the *CHEAP* can operate safely in an adversarial setting. We also conduct simulation-based comparative experiments in terms of diverse performance metrics. The experimental results imply that the *CHEAP* is a more efficient security protocol with smaller running time, less energy consumption, and lower communication overhead.**

***Index Terms*—Resource-Efficient, Data Type-Aware, Authentication, Chebyshev Polynomials, Internet of Things**

I. Introduction

Entering the third decade of the 21st century, the Internet of Things (IoT) technology has matured to support the fourth industrial revolution through becoming an inseparable component of modern businesses. IoT technology and its applications as a whole has been a topic of discussion for assisting with digital transformation recently. With the assistance of machine learning and edge computing techniques, IoT platforms are becoming the leading source of massive real-time data generated by critical systems such as digital twins, immersive/interactive environments, etc. According to the recent report of Statista, 74 zettabytes data will be generated by 19.08 billion IoT devices from all types of industry verticals and consumer markets globally in 2025 [1]. With good quality data, obtained with a clear use of IoT technology in mind, it is possible to get work done faster and more efficient with less effort.

Looking from the other side, the IoT paradigm opens the gate for a wide range of security risks and challenges to IoT devices, operation platforms, communication systems, and even the networks/systems to which they are connected. As new IoT applications continuously emerge and millions of heterogeneous IoT devices are connected to the Internet, IoT security and privacy has aroused people's wide concern. For instance, the number of documented IoT cyber attacks exceeded 111 millions in 2022 globally. The education and research sector took a serious hit with 131 weekly attacks targeting IoT devices per organization; that is more than twice the global average and a staggering 34% increase from the year before. Obviously, before reaping the fruits of IoT technology, it is extremely critical to protect IoT devices and systems from security attacks in the cyber-threat environment.

General speaking, IoT devices are equipped with application-specific sensors that continually gather and react to various data of surroundings (e.g., individuals, business operations, etc). As the IoT applications might be associated with sensitive and critical information, disclosing/compromising IoT device observation to/by unauthorized entities during transmission might compromise the objectives of IoT applications [2]. For example, e-health IoT systems require patients and doctors to communicate through a public wireless channel, as a result, either message falsification or doctor impersonator can put patients at risk [3]. In addition, IoT devices are regarded as embedded systems and are manufactured with small circuit with limited computing power and energy supply. Even though IoT manufacturers always claim that the batteries of their IoT devices have a shelf life of around ten years [4], however, demanding tasks or energy-hungry algorithms will make the batteries last only a small fraction of that time [5]. Last but not least, IoT devices might be deployed for different applications (different data type and sensitivity), e.g., smart hospital systems enable IoT devices to monitor room temperatures (non-sensitive data) and track the movement of patients with special needs (sensitive data) [6]. Encrypting all sensitive and non-sensitive data collected by the IoT device with the same secret session key might provide a chance for an unauthorized entity to access sensitive data which is meant for another authorized entity. In view of the above-mentioned research

challenges, it is extremely difficult to design suitable security protocols for IoT devices and systems.

During past years, several representative authentication protocols such as physically uncloneable function (PUF)-based [7], firmware-based [8], RFID-based [9], blockchain-based [10], and cross domain-based [11] solutions have been investigated to help IoT devices and IoT gateway exchange data over insecure communication channels. However, these existing solutions either rely on additional hardware (e.g., RFID [9]), incur high communication overhead (e.g., frequent communication with blockchain network [10]), or have intrinsic security flaws in their design (e.g., vulnerable to physical probing attack [11]). Especially, none of existing solutions distinguish between the different types of data collected by IoT devices during the authentication process, which will lead to data leakage that sensitive data are being accessed by unauthorized entities. Thus, what has been lacking in the current theory is a novel data type-aware authentication protocol that adopts resource-friendly computing operations to achieve the security, privacy, and performance requirements of IoT systems.

In this paper, we propose a resource-efficient and data type-aware authentication protocol using Chebyshev polynomials, hereafter referred to as *CHEAP*, for IoT systems. The *CHEAP* consists of two sub-schemes: (i) authentication and key establishment between IoT device and IoT gateway; and (ii) authentication and key establishment between two IoT devices. In order to evaluate *CHEAP*'s security properties and performance, we first use HLPSL modular language [12] to implement *CHEAP* and perform security verification on AVISPA [13]. After that, we conduct experiments in a simulation-based environment, where the *CHEAP*, SAE [14], and REAP [15] are implemented in Java and compared in terms of diverse performance metrics. The experimental results imply that the *CHEAP* is a safe security protocol with smaller running time, less energy consumption, and lower communication overhead, compared to existing schemes.

II. Related Work

In [16], the authors propose a key distribution and authentication architecture based on fog computing solution for e-health applications. In order to achieve authentication between the user and the fog server, there are three phases to proceed: initialization, registration, and authentication. First, the cloud service provider finalizes system parameters, bilinear pairing primitive, and hash function. Next, the fog server and the user complete registration with the cloud service provider after obtaining their cryptographic secrets. Finally, the user and the fog server exchange three messages to achieve mutual authentication through bilinear pairing and set up a secure session key using hash function. Even though the proposed authentication framework meets the pre-defined security requirements, however, bilinear pairing incurs a substantial computation overhead. In [17], a symmetric key authentication protocol (SKAFS) using PUF is proposed for IoT-edge-cloud computing systems. The SKAFS considers to adopt two kinds of PUFs: weak PUF and re-configurable PUF. Here, the weak PUF stores the long-term secret key of IoT device, while the re-configurable PUF is used to produce temporary secret key. The SKAFS shows promising performance, however, the adoption of two PUFs definitely increases the operational cost. In addition, the SKAFS is vulnerable to physical probing attack because the PUF stores the secret key directly.

The authors in [8] propose an authentication protocol for IoT devices based on the application code of firmware, where the secret cryptographic information of IoT device is calculated based on the firmware. If the generated cryptographic information is not matched with the information stored by the server, the authentication request of IoT device is rejected. Their approach is able to defend against device impersonation attack because of firmware integrity, however, the integration between authentication program and the application code of firmware is extremely challenging. In addition, the application code of firmware also has to be modified, which might compromise the integrity of firmware. In [18], a security analysis is conducted to disclose the security vulnerabilities of [19] such as untraceability and impersonation attack. Motivated by these security design flaws, a multi-factor authentication protocol using elliptic curve cryptography is proposed for IoT systems. However, the contribution of [18] is very limited, which follows the protocol framework of [19], but makes several operation remedies.

In summary, the security and privacy issues of IoT systems have received significant attention recently, and a wide range of authentication protocols have been designed to protect the communication between IoT entities. However, little effort has been made to develop data type-aware authentication protocol using lightweight techniques to meet the security, privacy, and performance requirements of IoT systems. The realization of such a protocol would be unprecedented because the similar technique is not currently available in the IoT community, and the proposed work will fill this research gap.

III. Network and Adversarial Models

A. Network Model

It is commonly agreed that IoT devices have limited processing power, storage space, as well as energy supply, and the IoT communication links are featured with low data rate and high channel error probabilities [20]. In order to efficiently facilitate data acquisition and transmission in resource and communication constrained IoT networks, RPL routing protocol [21] has been introduced as excellent supports of IoT technology. Thus, in this paper we assume that our network model is built based on RPL routing protocol. As shown in Fig. 1, the IoT network is organized into a tree-shaped structure, and this independent structure is officially called destination oriented directed acyclic graph, which is widely known as DODAG. In the DODAG, there are a set of IoT devices and one IoT gateway. The IoT devices are designed with the built-in ideal physical unclonable function (PUF) primitive so that the secret information can be dynamically calculated, rather than being cached directly in the storage unit. In the network, the IoT gateway is responsible for collecting

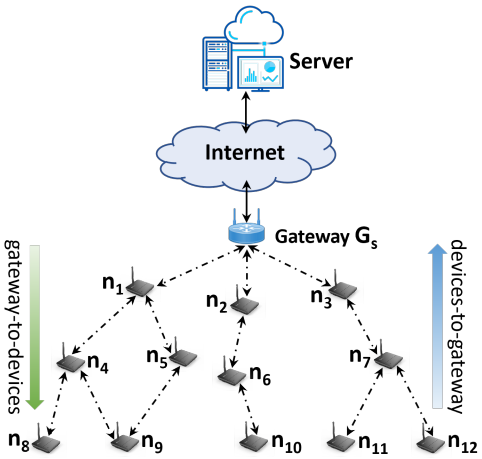


Fig. 1. Network model.

data from IoT devices and relaying them to the application server through the Internet. In order to efficiently collect data from IoT devices and distribute IoT gateway control messages, RPL routing protocol provides three communication modes, which are device-to-device, gateway-to-devices, and devices-to-gateway. Take devices-to-gateway communication model as an example, if the IoT device (e.g., n_1) has direct connection with the IoT gateway (e.g., G_s), it can directly send its data to the IoT gateway. Otherwise, the IoT devices (e.g., n_4 and n_{10}) need to send data to their preferred parent (e.g., n_1 is the preferred parent of n_4, and n_6 is the preferred parent of n_{10}) who will further relay data to their corresponding preferred nodes (e.g., n_2 is the preferred parent of n_6). This process will continue until the data reach the IoT gateway. Since the data relay and the control message forwarding are conducted through wide-open wireless channels, the identities of communication entities (e.g., IoT devices and gateway) should be verified along with the establishment of secure session key in advance.

B. Adversarial Model and Security Requirements

In this paper, the Dolev-Yao model [22] is adopted to shape the malicious behavior of adversaries. From an adversarial point of view, the attackers attempt to conduct stealthily eavesdropping with the hope of retrieving any useful secret information (e.g., cryptographic value or session key) that can be used to compromise the communication. As attackers, they can easily eavesdrop the communication of IoT entities because the wireless channel is an wide-open medium. However, the stealthily eavesdropping will end in failure if the IoT entities are able to exchange data or control messages over an encrypted channel. In addition, IoT applications might be deployed in unattended public areas without any protection, which makes IoT devices vulnerable to physical attacks. As a result, there is a chance for the opponent to physically approach to IoT devices and attempt to retrieve critical cryptographic information to compromise the future communication. But the adversarial attempt might not succeed at all. This is because IoT devices are built with the physical protection feature, which is PUF primitive, the physical probing attack will change or even destroy the PUF mapping which causes the same response not being re-generated with the same challenge. Thus, it is reasonable to assume that a maliciously compromised and re-programmed IoT device does not exist in the network. Other attacks such as jamming attacks can also be launched by the adversary, however, they are outside the scope of this paper. In addition, we assume that the IoT gateway is also a trusted entity in the network.

We design *CHEAP* to meet the following security requirements. First, the *CHEAP* shall achieve mutual authentication and session key establishment between IoT device and IoT gateway, and between two IoT devices. Second, in the *CHEAP* the pseudonym of IoT devices, rather than the real identities, will be used for the communication. Third, the *CHEAP* shall make sure that the messages are exchanged confidentially and the content of messages cannot be altered in transit. Finally, the *CHEAP* shall be secure against IoT device impersonation attack, IoT gateway impersonation attack, message modification attack, physical probing attack, replay attack, and man-in-the-middle attack.

IV. The Proposed Authentication Protocol

In this section, we present the proposed resource-efficient and data type-aware authentication protocol using Chebyshev polynomials, also called *CHEAP*, for IoT systems. The *CHEAP* realizes mutual authentication and session key establishment for the following two communication scenarios: (i) the IoT device attempts to communicate with the IoT gateway; and (ii) one IoT device attempts to communicate with another IoT device.

A. System Initialization

During the system initialization phase, the IoT gateway G_s on behalf of the IoT server initializes system parameters and chooses public functions.

1) G_s chooses a Chebyshev polynomial function $T_{(x)}(y)$, where x is an integer and $y \in$ [-1, 1].
2) G_s selects a secure one-way hash function H, where H:$\{0, 1\}^* \rightarrow \{0, 1\}^m$, where m indicates the number of bits.
3) G_s obtains its private key PR_s^G, and computes the corresponding public key $PU_s^G = T_{(PR_s^G)}(y)$. G_s stores PR_s^G safely.
4) G_s announces system parameters and functions $\{T_{(x)}(y), x, y, H, PU_s^G\}$.

B. IoT Device Registration

During the registration phase, IoT devices in the network register with the IoT gateway G_s. Without loss of generality, the IoT device n_i is chosen to demonstrate the registration process.

1) n_i picks up its unique real identification n_i and randomly chooses its PUF challenge che_i.
2) n_i retrieves its PUF response res_i with che_i and PUF F_i^{puf}, $res_i = F_i^{puf}(che_i)$.

3) n_i calculates its fake identification or pseudonym pn_i^{ts} with n_i, res_i and ts, pn_i^{ts} = $H(n_i \parallel res_i \parallel ts)$. Here, ts is the current system time.
4) n_i computes the public tag PT_i with res_i and $T_{(x)}(y)$, PT_i = $T_{(res_i)}(y)$.
5) n_i securely shares its real identification, pseudonym, PUF response, and public tag, $\{n_i, pn_i, res_i, PT_i\}$, with the IoT gateway G_s using the One-Time Password (OTP) algorithm [23].
6) G_s assigns n_i with a set of different tasks D_i = $[d_{i.1}, d_{i.2}, \cdots, d_{i.k}, \cdots, d_{i.z}]$ to complete, and shares D_i with n_i via a secure channel. Here, each task indicates different data type that n_i needs to collect and $d_{i.k}$ indicates the k^{th} task.
7) n_i stores n_i, pn_i^{ts}, che_i, PT_i, D_i, and PU_s^G in the memory, and discards res_i.
8) G_s adds the identity related information of n_i, $\{n_i, pn_i, res_i, PT_i, D_i\}$, in the database.

Since G_s is a trusted entity, then the database is assumed to be well protected. For security reasons, the IoT device n_i does not directly store res_i in the storage space. This is because res_i is used to calculate pn_i^{ts} and PT_i, and is considered as the critical cryptographic parameter. In addition, the pseudonym pn_i^{ts} is computed with the current system time ts to frequently update the pseudonym of IoT device n_i.

C. Authentication Between IoT Device and IoT Gateway

In this phase, the IoT device n_i and the IoT gateway G_s mutually validate each other's identifications and negotiate a secure session key for the follow-up communication of data type $d_{i.k}$.

1) n_i obtains its PUF challenge che_i from the memory and calculates the corresponding PUF response res_i = $F_i^{puf}(che_i)$.
2) n_i computes its public tag PT_i with res_i and $T_{(x)}(y)$, PT_i = $T_{(res_i)}(y)$.
3) n_i retrieves its real identification n_i, and calculates a new pseudonym $pn_i^{ts+\varpi}$ = $H(n_i \parallel res_i \parallel ts{+}\varpi)$, where $ts{+}\varpi$ is the current system time.
4) n_i randomly selects a number r_i and calculates the public Chebyshev polynomial PU_i^{pol} = $T_{(r_i \cdot res_i \cdot n_i \cdot d_{i.k})}(y)$
5) n_i fetches the IoT gateway G_s's public key PU_s^G from the storage space, and computes the secret Chebyshev polynomial PR_i^{pol} = $T_{(r_i \cdot res_i \cdot n_i \cdot d_{i.k})}(PU_s^G)$.
6) n_i calculates the following

$$H_{i,1} = pn_i^{ts+\varpi} \oplus H(pn_i^{ts} \| n_i),$$

$$H_{i,2} = d_{i.k} \oplus H(pn_i^{ts} \| n_i \| pn_i^{ts+\varpi}),$$

$$H_{i,3} = r_i \oplus H(pn_i^{ts} \| n_i \| pn_i^{ts+\varpi} \| d_{i.k}),$$

$$H_{i,4} = H(pn_i^{ts} \| n_i \| pn_i^{ts+\varpi} \| d_{i.k} \| r_i \| PR_i^{pol}).$$

7) n_i sends the authentication request message req_i = $\{pn_i^{ts}, H_{i,1}, H_{i,2}, H_{i,3}, H_{i,4}, ts{+}\varpi, PU_i^{pol}\}$ to G_s along the upward forwarding path.
8) G_s uses the IoT device n_i's old pseudonym $pn_i^{ts'}$ to retrieve n_i's identity related information from database, and recalculates n_i's new pseudonym $pn_i^{ts+\varpi'}$ = $H(n_i' \parallel res_i' \parallel ts{+}\varpi')$.
9) G_s computes the following, $pn_i^{ts+\varpi''}$ = $H_{i,1}' \oplus H(pn_i^{ts'} \parallel n_i')$, and checks whether $pn_i^{ts+\varpi'} \stackrel{?}{=} pn_i^{ts+\varpi''}$. If they are not equal, the authentication process is terminated. Otherwise, G_s calculates the following, $d_{i.k}'$ = $H_{i,2}' \oplus H(pn_i^{ts'} \parallel pn_i^{ts+\varpi'} \parallel n_i')$. G_s checks whether the data type $d_{i.k}'$ is the assigned data type in D_i'. If not, the authentication process is terminated. Otherwise, G_s proceeds with the following steps.
10) G_s computes PR_s^{pol} = $T_{(PR_s^G)}(PU_i^{pol'})$, where PR_s^{pol} = PR_i^{pol}. The proof is as follows,

$$\begin{aligned} PR_s^{pol} &= T_{(PR_s^G)}(PU_i^{pol'}) \\ &= T_{(PR_s^G)}(T_{(r_i \cdot res_i \cdot n_i \cdot d_{i.k})}(y)) \\ &= T_{(r_i \cdot res_i \cdot n_i \cdot d_{i.k})}(T_{(PR_s^G)}(y)) \\ &= T_{(r_i \cdot res_i \cdot n_i \cdot d_{i.k})}(PU_s^G) \\ &= PR_i^{pol}. \end{aligned}$$

In addition, G_s computes the following

$$r_i' = H_{i,3}' \oplus H(pn_i^{ts'} \| n_i' \| pn_i^{ts+\varpi'} \| d_{i.k}'),$$

$$H_{i,4}' = H(pn_i^{ts'} \| n_i' \| pn_i^{ts+\varpi'} \| d_{i.k}' \| r_i' \| PR_s^{pol}).$$

If $H_{i,4}' \neq H_{i,4}$, the authentication process is terminated. Otherwise, G_s proceeds with the following steps.
11) G_s arbitrarily chooses a random number r_u and calculates the public Chebyshev polynomial PU_s^{pol} = $T_{(r_u \cdot PR_s^G \cdot G_s)}(PT_i')$.
12) G_s calculates $H_{s,1}$ = $H(G_s \parallel pn_i^{ts+\varpi'} \parallel d_{i.k}' \parallel r_i' \parallel PU_s^{pol})$, and sends the authentication response message rep_i = $\{pn_i^{ts}, PU_s^{pol}, H_{s,1}\}$ to n_i along the downward forwarding path.
13) G_s computes the secure session key $SK_{s,i}$ = $T_{(r_u \cdot PR_s^G \cdot G_s)}(PU_i^{pol'})$, and updates n_i's new pseudonym $pn_i^{ts+\varpi'}$ in the database.
14) n_i recalculates $H_{s,1}'$ = $H(G_s \parallel pn_i^{ts+\varpi} \parallel d_{i.k} \parallel r_i \parallel PU_s^{pol'})$, and verifies $H_{s,1}' \stackrel{?}{=} H_{s,1}$. If the verification fails, n_i discards rep_i. Otherwise, n_i calculates the secure session key $SK_{i,s}$ = $T_{(r_i \cdot n_i \cdot d_{i.k})}(PU_s^{pol'})$. Here

$$\begin{aligned} SK_{i,s} &= T_{(r_i \cdot n_i \cdot d_{i.k})}(PU_s^{pol'}) \\ &= T_{(r_i \cdot n_i \cdot d_{i.k})}(T_{(r_u \cdot PR_s^G \cdot G_s)}(PT_i)) \\ &= T_{(r_u \cdot PR_s^G \cdot G_s)}(T_{(r_i \cdot n_i \cdot d_{i.k})}(PT_i)) \\ &= T_{(r_u \cdot PR_s^G \cdot G_s)}(T_{(r_i \cdot n_i \cdot d_{i.k})}(T_{(res_i)}(y))) \\ &= T_{(r_u \cdot PR_s^G \cdot G_s)}(T_{(r_i \cdot res_i \cdot n_i \cdot d_{i.k})}(y)) \\ &= T_{(r_u \cdot PR_s^G \cdot G_s)}(PU_i^{pol}) \\ &= SK_{s,i}. \end{aligned}$$

At this moment, the IoT device n_i and the IoT gateway G_s have completed the mutual authentication and set up a

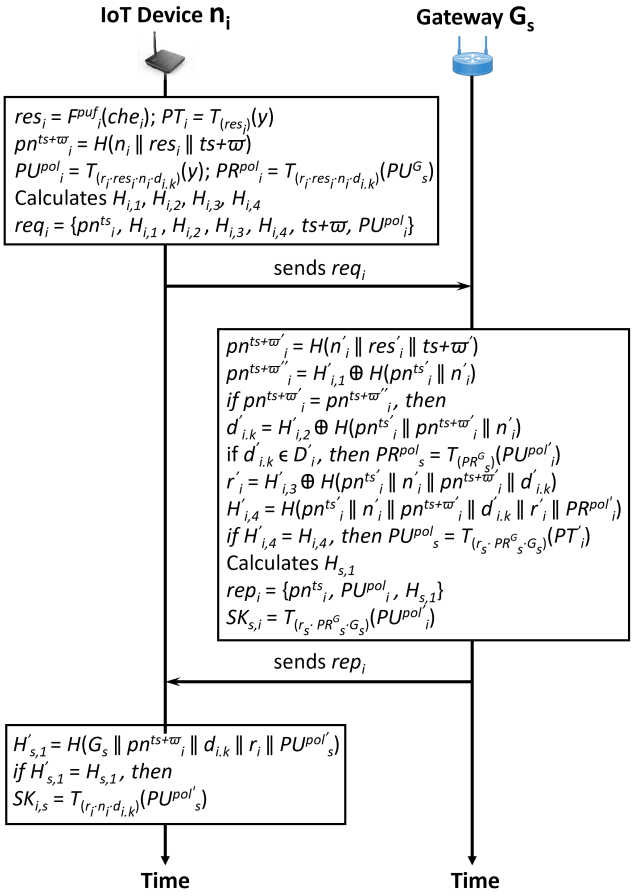


Fig. 2. The process of mutual authentication and key establishment between IoT device and IoT gateway.

secure session key for the data type $d_{i.k}$. The process of authentication and key establishment for the communication between IoT device and IoT gateway is shown in Fig. 2.

D. Authentication Between Two IoT Devices

In this phase, the IoT device n_i and the IoT device n_j mutually validate each other's identifications and obtain a secure session key with the assistance of the IoT gateway G_s for the follow-up communication of data type d_k. Here, both n_i and n_j are registered for the data type d_k.

1) n_i obtains its PUF challenge che_i from the memory and calculates the corresponding PUF response $res_i = F^{puf}_i(che_i)$.
2) n_i computes its public tag PT_i with res_i and $T_{(x)}(y)$, $PT_i = T_{(res_i)}(y)$.
3) n_i retrieves its real identification n_i, and calculates a new pseudonym $pn^{ts+\varpi}_i = H(n_i \| res_i \| ts+\varpi)$, where $ts+\varpi$ is the current system time.
4) n_i randomly selects a number r_i and calculates the public Chebyshev polynomial $PU^{pol}_i = T_{(r_i \cdot res_i \cdot n_i \cdot d_k)}(y)$
5) n_i fetches the IoT gateway G_s's public key PU^G_s from the storage space, and computes the secret Chebyshev polynomial $PR^{pol}_i = T_{(r_i \cdot res_i \cdot n_i \cdot d_k)}(PU^G_s)$.
6) n_i calculates the following

$$H_{i,1} = pn^{ts+\varpi}_i \oplus H(pn^{ts}_j \| pn^{ts}_i \| n_i),$$

$$H_{i,2} = d_k \oplus H(pn^{ts}_j \| pn^{ts}_i \| n_i \| pn^{ts+\varpi}_i),$$

$$H_{i,3} = r_i \oplus H(pn^{ts}_j \| pn^{ts}_i \| n_i \| pn^{ts+\varpi}_i \| d_k),$$

$$H_{i,4} = H(pn^{ts}_j \| pn^{ts}_i \| n_i \| pn^{ts+\varpi}_i \| d_k \| r_i \| PR^{pol}_i).$$

7) n_i sends the authentication request message $req_{i.s} = \{pn^{ts}_i, pn^{ts}_j, H_{i,1}, H_{i,2}, H_{i,3}, H_{i,4}, ts+\varpi, PU^{pol}_i\}$ to G_s along the upward forwarding path. Here, pn^{ts}_j is the pseudonym of IoT device n_j.
8) G_s uses the IoT device n_i's and n_j's old pseudonym $pn^{ts'}_i$ and $pn^{ts'}_j$ to retrieve n_i's and n_j's identity related information from database respectively, and recalculates n_i's new pseudonym $pn^{ts+\varpi'}_i = H(n'_i \| res'_i \| ts+\varpi')$.
9) G_s computes the following, $pn^{ts+\varpi''}_i = H'_{i,1} \oplus H(pn^{ts'}_j \| pn^{ts'}_i \| n'_i)$, and checks whether $pn^{ts+\varpi'}_i \stackrel{?}{=} pn^{ts+\varpi''}_i$. If they are not equal, the authentication process is terminated. Otherwise, G_s calculates the following, $d'_k = H'_{i,2} \oplus H(pn^{ts'}_j \| pn^{ts'}_i \| pn^{ts+\varpi'}_i \| n'_i)$. G_s checks whether d'_k is the assigned data type in both D'_i and D'_j. If not, the authentication process is terminated. If yes, G_s proceeds with the following steps.
10) G_s computes $PR^{pol}_s = T_{(PR^G_s)}(PU^{pol'}_i)$, where $PR^{pol}_s = PR^{pol}_i$. The proof is as follows,

$$\begin{aligned} PR^{pol}_s &= T_{(PR^G_s)}(PU^{pol'}_i) \\ &= T_{(PR^G_s)}(T_{(r_i \cdot res_i \cdot n_i \cdot d_k)}(y)) \\ &= T_{(r_i \cdot res_i \cdot n_i \cdot d_k)}(T_{(PR^G_s)}(y)) \\ &= T_{(r_i \cdot res_i \cdot n_i \cdot d_k)}(PU^G_s) \\ &= PR^{pol}_i. \end{aligned}$$

In addition, G_s computes the following

$$r'_i = H'_{i,3} \oplus H(pn^{ts'}_j \| pn^{ts'}_i \| n'_i \| pn^{ts+\varpi'}_i \| d'_k),$$

$$H'_{i,4} = H(pn^{ts'}_j \| pn^{ts'}_i \| n'_i \| pn^{ts+\varpi'}_i \| d'_k \| r'_i \| PR^{pol}_s).$$

If $H'_{i,4} \neq H_{i,4}$, the authentication process is terminated. Otherwise, G_s proceeds with the following steps.
11) G_s calculates $H_{s,1} = H(pn^{ts'}_i \| pn^{ts'}_j \| n'_j \| d'_k \| res'_j \| ts+\varrho)$, and sends the authentication request message $req_{s.j} = \{pn^{ts'}_j, pn^{ts'}_i, H_{s,1}, ts+\varrho\}$ to n_j along the downward forwarding path.
12) n_j checks whether $ts+\varrho'$ is current, and computes $H'_{s,1} = H(pn^{ts''}_i \| pn^{ts}_j \| n_j \| d_k \| res_j \| ts+\varrho')$. If $H'_{s,1} \neq H_{s,1}$, n_j discards rep_s. Otherwise, n_j proceeds with the following steps.
13) n_j obtains its PUF challenge che_j from the memory and calculates the corresponding PUF response $res_j = F^{puf}_j(che_j)$.
14) n_j computes its public tag PT_j with res_j and $T_{(x)}(y)$, $PT_j = T_{(res_j)}(y)$.
15) n_j retrieves its real identification n_j, and calculates a new pseudonym $pn^{ts+\mu}_j = H(n_j \| res_j \| ts+\mu)$, where $ts+\mu$ is the current system time, and $ts+\mu > ts+\varrho$.

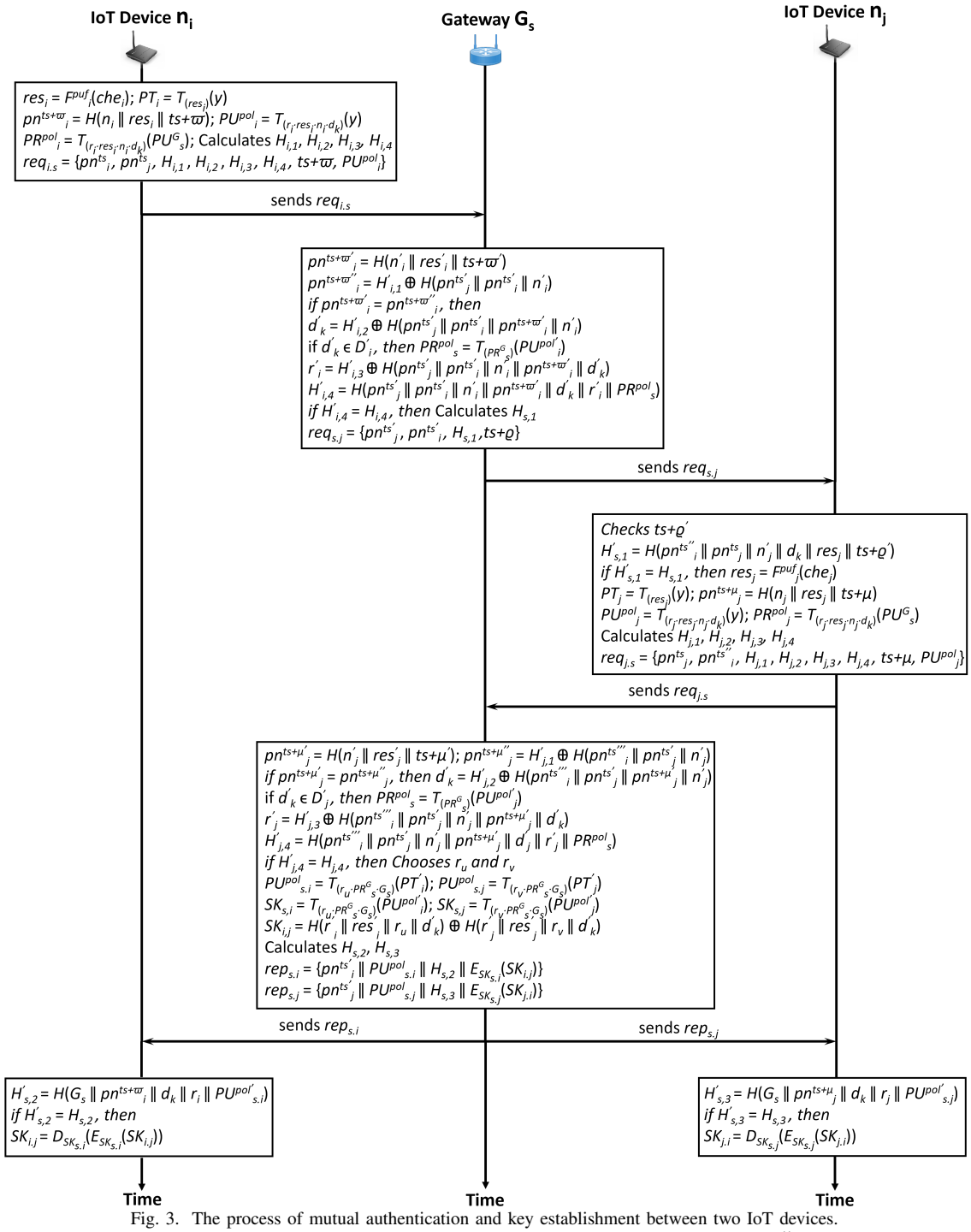


Fig. 3. The process of mutual authentication and key establishment between two IoT devices.

16) n_j randomly selects a number r_j and calculates the public Chebyshev polynomial $PU_j^{pol} = T_{(r_j \cdot res_j \cdot n_j \cdot d_k)}(y)$
17) n_j fetches the IoT gateway G_s's public key PU_s^G from the storage space, and computes the secret Chebyshev polynomial $PR_j^{pol} = T_{(r_j \cdot res_j \cdot n_j \cdot d_k)}(PU_s^G)$.
18) n_j calculates the following

$$H_{j,1} = pn_j^{ts+\mu} \oplus H(pn_i^{ts''} \| pn_j^{ts} \| n_j),$$

$$H_{j,2} = d_k \oplus H(pn_i^{ts''} \| pn_j^{ts} \| n_j \| pn_j^{ts+\mu}),$$

$$H_{j,3} = r_j \oplus H(pn_i^{ts''} \| pn_j^{ts} \| n_j \| pn_j^{ts+\mu} \| d_k),$$

$$H_{j,4} = H(pn_i^{ts''} \| pn_j^{ts} \| n_j \| pn_j^{ts+\mu} \| d_k \| r_j \| PR_j^{pol}).$$

19) n_j sends the authentication request message $req_{j.s} = \{pn_j^{ts}, pn_i^{ts''}, H_{j,1}, H_{j,2}, H_{j,3}, H_{j,4}, ts+\mu, PU_j^{pol}\}$ to G_s along the upward forwarding path.
20) G_s uses the IoT device n_j's old pseudonym $pn_j^{ts'}$ to retrieve n_j's identity related information from database,

and recalculates n_j's new pseudonym $pn_j^{ts+\mu^{'}}$ = $H(n_j^{'} \parallel res_j^{'} \parallel ts+\mu^{'})$.

21) G_s computes the following, $pn_j^{ts+\mu^{''}} = H_{j,1}^{'} \oplus H(pn_i^{ts^{'''}} \parallel pn_j^{ts^{'}} \parallel n_j^{'})$, and checks whether $pn_j^{ts+\mu^{'}} \stackrel{?}{=} pn_j^{ts+\mu^{''}}$. If they are not equal, the authentication process is terminated. Otherwise, G_s calculates $d_k^{'} = H_{j,2}^{'} \oplus H(pn_i^{ts^{'''}} \parallel pn_j^{ts^{'}} \parallel pn_j^{ts+\mu^{'}} \parallel n_j^{'})$. G_s checks whether $d_k^{'}$ is the assigned data type in $D_j^{'}$. If not, the authentication process is terminated. If yes, G_s proceeds with the following steps.

22) G_s computes $PR_s^{pol} = T_{(PR_s^G)}(PU_j^{pol^{'}})$, where $PR_s^{pol} = PR_j^{pol}$. The proof is as follows,

$$\begin{aligned} PR_s^{pol} &= T_{(PR_s^G)}(PU_j^{pol^{'}}) \\ &= T_{(PR_s^G)}(T_{(r_j \cdot res_j \cdot n_j \cdot d_k)}(y)) \\ &= T_{(r_j \cdot res_j \cdot n_j \cdot d_k)}(T_{(PR_s^G)}(y)) \\ &= T_{(r_j \cdot res_j \cdot n_j \cdot d_k)}(PU_s^G) \\ &= PR_j^{pol}. \end{aligned}$$

In addition, G_s computes the following

$$r_j^{'} = H_{i,3}^{'} \oplus H(pn_i^{ts^{'''}} \| pn_j^{ts^{'}} \| n_j^{'} \| pn_j^{ts+\mu^{'}} \| d_k^{'}),$$

$$H_{j,4}^{'} = H(pn_i^{ts^{'''}} \| pn_j^{ts^{'}} \| n_j^{'} \| pn_j^{ts+\mu^{'}} \| d_k^{'} \| r_j^{'} \| PR_s^{pol}).$$

If $H_{j,4}^{'} \neq H_{j,4}$, the authentication process is terminated. Otherwise, G_s proceeds with the following steps.

23) G_s arbitrarily chooses two random numbers, r_u and r_v, and calculates two public Chebyshev polynomials, $PU_{s.i}^{pol} = T_{(r_u \cdot PR_s^G \cdot G_s)}(PT_i^{'})$ and $PU_{s.j}^{pol} = T_{(r_v \cdot PR_s^G \cdot G_s)}(PT_j^{'})$, for n_i and n_j, respectively,

24) G_s computes the secure session key for n_i and n_j, $SK_{s.i} = T_{(r_u \cdot PR_s^G \cdot G_s)}(PU_i^{pol^{'}})$ and $SK_{s.j} = T_{(r_v \cdot PR_s^G \cdot G_s)}(PU_j^{pol^{'}})$, and updates n_i's and n_j's new pseudonym $pn_i^{ts+\varpi^{'}}$ and $pn_j^{ts+\varpi^{'}}$ in the database, respectively.

25) G_s calculates the secure session key for the communication between n_i and n_j as $SK_{i.j}$ (or $SK_{j.i}$) = $H(r_i^{'} \parallel res_i^{'} \parallel r_u \parallel d_k^{'}) \oplus H(r_j^{'} \parallel res_j^{'} \parallel r_v \parallel d_k^{'})$.

26) G_s calculates the following

$$H_{s,2} = H(G_s \| pn_i^{ts+\varpi^{'}} \| d_k^{'} \| r_i^{'} \| PU_{s.i}^{pol}),$$

$$H_{s,3} = H(G_s \| pn_j^{ts+\mu^{'}} \| d_k^{'} \| r_j^{'} \| PU_{s.j}^{pol}),$$

and sends the authentication response message $rep_{s.i} = \{pn_i^{ts^{'}}, PU_{s.i}^{pol}, H_{s,2}, E_{SK_{s.i}}(SK_{i.j})\}$ and $rep_{s.j} = \{pn_j^{ts^{'}}, PU_{s.j}^{pol}, H_{s,3}, E_{SK_{s.j}}(SK_{j.i})\}$ to n_i and n_j along the downward forwarding path, respectively. Here, $E_{SK_{a.b}}(C)$ is an encryption function to encrypt the content C with the secret key $SK_{a.b}$.

27) n_i recalculates $H_{s,2}^{'} = H(G_s \parallel pn_i^{ts+\varpi} \parallel d_k \parallel r_i \parallel PU_{s.i}^{pol^{'}})$, and verifies $H_{s,2}^{'} \stackrel{?}{=} H_{s,2}$. If the verification fails, n_i discards $rep_{s.i}$. Otherwise, n_i calculates the secure session key $SK_{i,s} = T_{(r_i \cdot n_i \cdot d_k)}(PU_{s.i}^{pol^{'}})$. Here

$$\begin{aligned} SK_{i,s} &= T_{(r_i \cdot n_i \cdot d_k)}(PU_{s.i}^{pol^{'}}) \\ &= T_{(r_i \cdot n_i \cdot d_k)}(T_{(r_u \cdot PR_s^G \cdot G_s)}(PT_i)) \\ &= T_{(r_u \cdot PR_s^G \cdot G_s)}(T_{(r_i \cdot n_i \cdot d_k)}(PT_i)) \\ &= T_{(r_u \cdot PR_s^G \cdot G_s)}(T_{(r_i \cdot n_i \cdot d_k)}(T_{(res_i)}(y))) \\ &= T_{(r_u \cdot PR_s^G \cdot G_s)}(T_{(r_i \cdot res_i \cdot n_i \cdot d_k)}(y)) \\ &= T_{(r_u \cdot PR_s^G \cdot G_s)}(PU_i^{pol}) \\ &= SK_{s,i}. \end{aligned}$$

28) n_i use $SK_{i,s}$ to decrypt the encrypted secure session key with n_j, $SK_{i.j} = D_{SK_{i,s}}(E_{SK_{s.j}}(SK_{i.j}))$. Here, $D_{SK_{a.b}}(C)$ is an decryption function to decrypt the content C with the secret key $SK_{a.b}$.

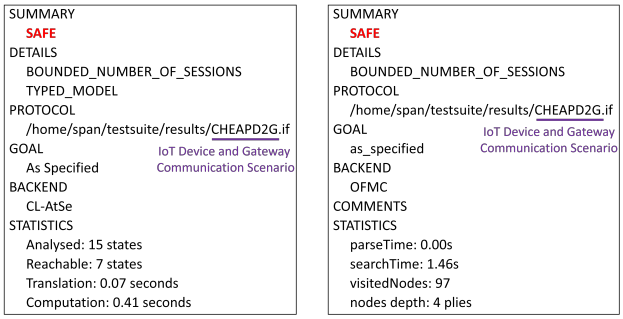
29) n_j performs the above similar operations to obtain the secure session key with n_i. The steps are ignored here.

Now the IoT device n_i and the IoT device n_j have set up a secure session key for the communication of data type d_k. The process of authentication and key establishment for the communication between two IoT devices is shown in Fig. 3.

V. Security Verification and Analysis

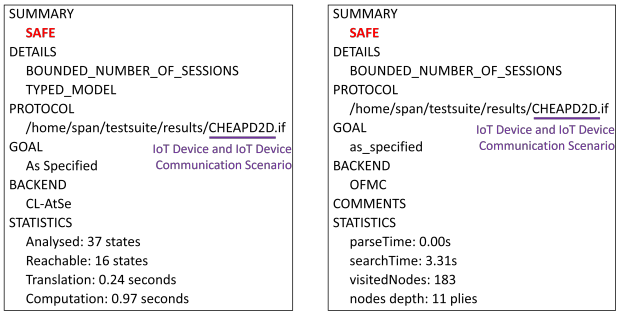
We use AVISPA [13], which is a specific tool to validate the security properties and design of communication protocols, to evaluate the security performance of *CHEAP* in the adversarial environment, where the attackers are launching man-in-the-middle, replay, and other cyber attacks. Recently AVISPA has gained popularity from cryptographic community because of the integrated modular language, easy-to-use features, user-friendly interface, and clear interpretable results. First of all, we set up a virtual machine environment using Virtual Box on a Windows PC, and launch the pre-built AVISPA virtual machine image [13]. Then, we implement *CHEAP* in HLPSL [12], which is a modular language integrated with AVISPA for the implementation and verification of communication protocols. In the HLPSL programs, we define two entity roles, e.g., IoT device and IoT gateway, one communication role, e.g., session, and three setting roles, e.g., adversary, protocol objective, and environment. Finally, we execute the HLPSL programs with CL-AtSe and OFMC back-ends which will expose the potential attacks and vulnerable scenarios of *CHEAP* in the format of communication sequence diagram. We did not select other two back-ends (e.g., SATMC and TA4SP) because they do not support bitwise XOR operation. As shown in Fig. 4 and 5, the *CHEAP* has passed all security testings of AVISPA and obtained ''SAFE'' certificates for both sub-schemes. In summary, the *CHEAP* is a safe cryptographic protocol without any security design flaws and has ability to defend against common cyber attacks.

In the following, we analyze the *CHEAP* and explain how it meets all pre-defined security requirements in Subsection III.C. First, the *CHEAP* can realize mutual authentication and session key establishment for the communication between IoT device and IoT gateway as well as the communication



(a) (b)

Fig. 4. Security verification results using AVISPA for the communication scenario between IoT device and IoT gateway.



(a) (b)

Fig. 5. Security verification results using AVISPA for the communication scenario between two IoT devices.

between two IoT devices. In Subsection. IV.C, the process of mutual authentication and session key establishment for the communication between IoT device and IoT gateway is presented. The IoT device first creates a set of hash values which contain its secret identity related information (e.g., real identification, PUF response, registered data type, randomly generated number), and then sends an authentication request message piggybacked with hash values and other information to the IoT gateway. With the pre-shared IoT device's identity information and the content of authentication request message, the IoT gateway is able to verify the real identity of IoT device. After that, the IoT gateway replies an authentication response message which contains the information to set up the secret session key with the IoT device. In addition, the *CHEAP* can assist with two IoT devices to negotiate a secret session key after identity verification. In Subsection. IV.D, the IoT gateway first verifies the identities of two IoT devices, and then sets up the secret session key on behalf of them. Thus, the *CHEAP* can realize mutual authentication and session key establishment for for the communication between IoT device and IoT gateway as well as the communication between two IoT devices. Second, in the *CHEAP* the IoT device creates a pseudonym with its real identification, PUF response, and timestamp, and uses the pseudonym to communicate with the IoT gateway. Thus, the anonymity of IoT devices can be provided by the *CHEAP*. Third, the *CHEAP* can maintain the confidentiality of critical information using hash function. For example, when the IoT device wants to establish a secure session key with the IoT gateway, it needs to send its secret identity related information (e.g., real identification, PUF response, registered data type) to the IoT gateway via an authentication request message. Instead of sending the identity related information in plaintext, the IoT device produces hash values to hide them in the authentication request message. Therefore, the *CHEAP* can guarantee that the messages are exchanged confidentially and the content of messages cannot be altered in transit.

TABLE I
SECURITY REQUIREMENTS

Security Requirement	*CHEAP*
Authentication Between IoT Device and IoT Gateway	✓
Authentication Between Two IoT Devices	✓
Data Type Aware Session Key Establishment	✓
Confidentiality / Integrity / Anonymity	✓
IoT Device / Gateway Impersonation Attack	✓
Message Modification Attack	✓
Physical Probing Attack	✓
Replay Attack	✓
Man-In-The-Middle Attack	✓

The *CHEAP* is secure against several common cyber attacks. First, the *CHEAP* does not have security design flaws that might be exploited by man-the-middle and replay attacks. This has been proved through the security verification in AVISPA, where the *CHEAP* passes all security testing in the adversarial environment. The *CHEAP* is safe from IoT device impersonation attack. Since the IoT device uses the PUF response for identity verification and the PUF response is uniquely generated with the IoT device's PUF and challenge, thus, the adversary cannot impersonate any IoT devices in the network. The *CHEAP* does not suffer from IoT gateway impersonation attack. The adversary might capture the exchanged authentication request message between IoT device and IoT gateway, however, it does not have the valid IoT device identity related information (e.g., real identification, PUF response) to decrypt the message. As a result, the adversary cannot impersonate as the IoT gateway to reply a valid authentication response message to the IoT device. The *CHEAP* can defend against message modification attack because the communication entity verifies the content of message using hash function. Thus, the modification of message content can be easily detected. The IoT devices in the networks are safe from physical probing attack. This is because the IoT devices dynamically calculate the cryptographic information, e.g., PUF response, instead of storing it directly in the memory space. When the adversary attempts to probe the integrate circuit of IoT device for the cryptographic information, the PUF will be destroyed and the same PUF response cannot be regenerated. Therefore, the *CHEAP* is immune to physical probing attack. The summary of satisfied security requirements is provided in Table I.

VI. PERFORMANCE EVALUATION

To evaluate the performance of *CHEAP*, an experimental environment is set up on the Windows PC where we conduct simulation-based experiments. The experimental environment is mainly just an integrated development platform, Eclipse IDE for Java. Thus, the program implementation only simulates the computation of security protocols, while the performance of wireless communication (e.g., message exchange between IoT entities) is measured in other ways. The Windows PC is

TABLE II
COMPARISON OF COMMUNICATION OVERHEAD

Scheme	No. of Rqd. Messages	Energy Consumption (Joule)
CHEAP		
*D2G Comm.	2	2.258135×10^{-4}
$^{\diamond}$D2D Comm.	5	5.645338×10^{-4}
SAE		
$^{\bullet}$D2G Comm.	9	10.161608×10^{-4}
$^{\blacklozenge}$D2D Comm.	10	11.290675×10^{-4}
REAP		
$^{\circ}$U2D Comm.	3	3.387203×10^{-4}
$^{\triangleright}$D2D Comm.	6	6.774405×10^{-4}

running Windows 10 Pro operating system and is powered by a 4^{th} generation Intel Core i5 processor with 6 MB cache and max 3.9 GHz turbo frequency. In order to distinguish similarities and differences between the *CHEAP* and the state of the art, a comparative study of three security protocols, e.g., *CHEAP*, SAE [14], and REAP [15], is made in the experimental environment. The basic idea of SAE is that the new IoT device sends an authentication request message to neighbor IoT devices who have already been authenticated to request to join the network. The authentication request message contains identity information of new IoT device and is encrypted with one of the pre-loaded secret keys. If the neighbor IoT device is able to decrypt the authentication request message and verify the identity of new IoT device, it sends a session key request message to the IoT gateway that will issue a secure session key for the new IoT device. The strength of SAE is to use already authenticated IoT devices to authenticate new IoT device, which can reduce the authentication overhead of IoT gateway. In the REAP, the user first goes through local identity verification using password and biometric, and then sends an authentication request message to the gateway. After the gateway verifies the pre-assigned secret parameters from the user, it communicates with the IoT device to establish a secure session key. The metrics chosen to demonstrate the performance of *CHEAP*, SAE [14], and REAP [15] are communication overhead, execution time, as well as energy consumption.

First, we measure the number of required messages to complete the process of authentication and key establishment, and then calculate the corresponding energy consumption of communication for *CHEAP*, SAE, and REAP. As shown in Table II, the *CHEAP* requires the smallest number of messages to achieve authentication and key establishment for the communication between IoT device and IoT gateway (e.g., D2G Comm.), and the communication between two IoT devices (e.g., D2D Comm.). According to Fig. 2, when the IoT device wants to communicate with the IoT gateway, it first sends an authentication request message to the IoT gateway. After the IoT gateway verifies the identity of IoT device, it replies an authentication response message to the IoT device to set up the session key. Thus, only two messages are required to achieve authentication and key establishment for the communication between IoT device and IoT gateway. As shown in Fig. 3, five messages are needed to achieve authentication and key establishment for the communication between two IoT

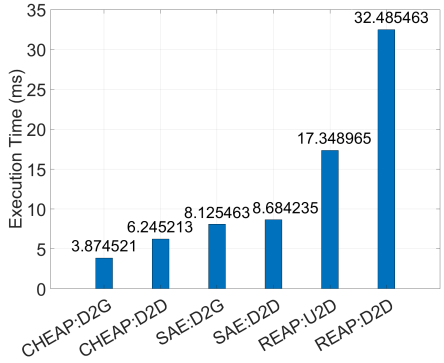


Fig. 6. Comparison of execution time.

devices. One IoT device first sends an authentication request message to the IoT gateway to request to communicate with another IoT device. Then, the IoT gateway forwards the authentication request message to another IoT device that will reply an authentication request message back. Finally, the IoT gateway sends an authentication response message to both IoT devices respectively to finish the key establishment process. Based on the communication sequence diagram presented in the SAE [14], we learn that the SAE would need nine and ten messages to achieve authentication and key establishment for the communication between IoT device and IoT gateway (e.g., D2G Comm.), and the communication between two IoT devices (e.g., D2D Comm.). The REAP requires less number of messages than SAE, where three messages are needed to successfully set up a secure session key between the user and the IoT device and six messages would be exchanged for two IoT devices setting up a secure session key. With the amount of exchanged messages, we also calculate the energy consumption of communication. Here, the energy consumption of communication is the product of the number of transmitted messages times the energy consumption of sending and receiving a single message [24]. Since the *CHEAP* has the smallest number of exchanged messages, it also consumes the least amount of energy for the process of authentication and key establishment compared to SAE and REAP.

Second, we measure the execution time of *CHEAP*, SAE, and REAP in terms of two communication scenarios in Fig. 6. Overall, the *CHEAP* has the lowest execution time compared to SAE and REAP. In the *CHEAP*, lightweight operations such as Chebyshev polynomial, PUF, and hash function are adopted to verify the identities of IoT entities and establish secure session keys for them to communicate. As a result, a significant amount of time can be reduced to execute the cryptographic operations of *CHEAP*, and the lowest execution time is obtained. Since the similar idea and operations are used to implement the process of authentication and key establishment for the communication between IoT device and IoT gateway, and the communication between two IoT devices, thus, the execution time of *CHEAP* is lower than that of SAE and REAP for the above two communication scenarios. The REAP has the longest execution time because of the complexity of authentication and key establishment process. In the REAP, the user is considered as one of communication

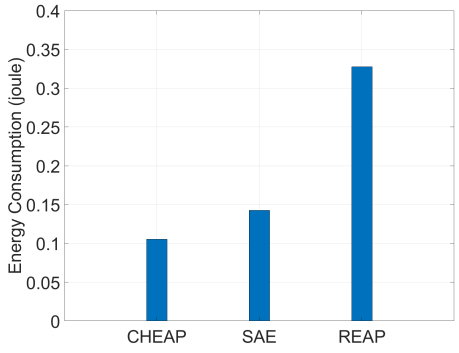


Fig. 7. Comparison of energy consumption.

entities. Before the user can communicate with the gateway, the REAP needs to verify the identity of user using password and biometric. And then, the REAP uses the biometric related results to generate several cryptographic numbers for the authentication with the gateway and the IoT device. Therefore, a longer time is required by REAP. The SAE takes less amount of time than REAP because a set of secret keys are pre-loaded into IoT devices. Thus, the IoT device can directly select one of keys to authenticate with other IoT devices.

Third, we obtain the energy consumption of executing *CHEAP*, SAE, and REAP. As shown in Fig. 7, the *CHEAP* consumes the smallest amount of energy while achieving authentication and key establishment for two communication scenarios. The rationale is that the *CHEAP* adopts resource-friendly operations which consume less amount of energy. The SAE consumes more energy than *CHEAP* because it frequently uses symmetric key encryption and decryption algorithms to produce the ciphertext of pre-loaded key and random numbers. The REAP consumes the largest amount of energy. This is because resource-hungry techniques such as biometric-based fuzzy extractor algorithms and additional encryption algorithm AEGIS are adopted in the protocol.

VII. Conclusion

In this paper, we put the effort into investigating the security and privacy issues of IoT systems, and then proposed a resource-efficient and data type-aware authentication protocol (*CHEAP*). The *CHEAP* is designed to achieve authentication and key establishment for the communication between IoT device and IoT gateway, and between two IoT devices. In addition, the secure session key is created based on the registered data type of IoT devices, so that an unauthorized IoT device will not be able to access sensitive data which is meant for another authorized IoT device. To evaluate the security properties and performance of *CHEAP*, we implemented *CHEAP* in HLPSL and performed a security verification on AVISPA. The security verification results indicate that the *CHEAP* is a secure protocol and does not have any security weaknesses. We also built an experimental simulation framework, implemented *CHEAP* and its counterparts in Java, and then evaluated and analyzed their performance in various metrics. The experimental results showed that the *CHEAP* is a more efficient security protocol.

References

[1] *Number of IoT Connected Devices Worldwide 2019-2021, With Forecasts to 2030*, https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[2] C. Pu, "Sybil Attack in RPL-Based Internet of Things: Analysis and Defenses," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4937–4949, 2021.

[3] C. Pu, H. Zerkle, A. Wall, S. Lim, K. Choo, and I. Ahmed, "A Lightweight and Anonymous Authentication and Key Agreement Protocol for Wireless Body Area Networks," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 21 136–21 146, 2022.

[4] *Get 10 Years From 9 Volts: The Power of LoRaWAN*, https://blogs.cisco.com/government/get-10-years-from-9-volts-the-power-of-lorawan.

[5] C. Pu and K. Choo, "Lightweight Sybil Attack Detection in IoT based on Bloom Filter and Physical Unclonable Function," *Elsevier Computers & Security*, vol. 113, p. 102541, 2022.

[6] A. Rathnayaka *et al.*, "An Autonomous IoT-based Contact Tracing Platform in a COVID-19 Patient Ward," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8706–8717, 2023.

[7] C. Pu, A. Wall, and K. Choo, "Bilinear Pairing and PUF Based Lightweight Authentication Protocol for IoD Environment," in *Proc. IEEE MASS*, 2022, pp. 115–121.

[8] Z. Chen, Z. Cheng, W. Luo, J. Ao, Y. Liu, K. Sheng, and L. Chen, "FSMFA: Efficient firmware-secure multi-factor authentication protocol for IoT devices," *Internet of Things Journal*, vol. 21, p. 100685, 2023.

[9] T. Li, Y. Liu, and J. Ning, "SDRLAP: A secure lightweight RFID mutual authentication protocol based on PUF with strong desynchronization resistance," *Peer-to-Peer Networking and Applications*, pp. 1–16, 2023.

[10] K. Qian, Y. Liu, X. He, M. Du, S. Zhang, and K. Wang, "HPCchain: A Consortium Blockchain System based on CPU-FPGA Hybrid PUF for Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, pp. 1–11, 2023.

[11] K. Wang, K. Sun, J. Dong, L. Sha, and F. Xiao, "AP-CDE: Cost-Efficient Authentication Protocol for Cross-Domain Data Exchange in IIoT," *IEEE Systems Journal*, pp. 1–12, 2023.

[12] Y. Chevalier *et al.*, "A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols," in *Proc. SAPS*, 2004, pp. 1–13.

[13] *SPAN*, http://www.avispa-project.org/.

[14] P. Rosa, A. Souto, and J. Cecílio, "Light-SAE: A Lightweight Authentication Protocol for Large-Scale IoT Environments Made with Constrained Devices," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.

[15] M. Tanveer, A. Alkhayyat, A. Khanand, N. Kumar, and A. Alharbi, "REAP-IIoT: Resource-Efficient Authentication Protocol for the Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 453–24 465, 2022.

[16] S. Mookherji, V. Odelu, R. Prasath, A. Das, and Y. Park, "Fog-Based Single Sign-On Authentication Protocol for Electronic Healthcare Applications," *IEEE Internet of Things Journal*, vol. 10, no. 12, pp. 10 983–10 996, 2023.

[17] M. Seifelnasr, R. AlTawy, and A. Youssef, "SKAFS: Symmetric Key Authentication Protocol with Forward Secrecy for Edge Computing," *IEEE Internet of Things Journal*, pp. 1–1, 2023.

[18] Y. Li, "A secure and efficient three-factor authentication protocol for IoT environments," *Journal of Parallel and Distributed Computing*, vol. 179, p. 104714, 2023.

[19] A. Mirsaraei, A. Barati, and H. Barati, "A secure three-factor authentication scheme for IoT environments," *Journal of Parallel and Distributed Computing*, vol. 169, pp. 87–105, 2022.

[20] C. Pu, J. Brown, and L. Carpenter, "A Theil Index-Based Countermeasure Against Advanced Vampire Attack in Internet of Things," in *Proc. IEEE HPSR*, 2020, pp. 1–6.

[21] T. Winter *et al.*, *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, 2012.

[22] Q. Do, B. Martini, and K. Choo, "The role of the adversary model in applied security research," *Computers & Security*, vol. 81, pp. 156–181, 2019.

[23] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, *TOTP: Time-Based One-Time Password Algorithm*, 2011.

[24] C. Pu and S. Lim, "A Light-Weight Countermeasure to Forwarding Misbehavior in Wireless Sensor Networks: Design, Analysis, and Evaluation," *IEEE Systems Journal*, vol. 12, no. 1, pp. 834–842, 2018.