Privacy-Preserving Mutual Authentication Protocol With Forward Secrecy for IoT–Edge–Cloud

Mohamed Seifelnasr, *Student Member, IEEE*, Riham AlTawy, *Senior Member, IEEE*, Amr Youssef, *Senior Member, IEEE*, and Essam Ghadafi, *Senior Member, IEEE*

***Abstract*—The three-tier IoT–Edge–Cloud paradigm enables low-end devices to use the computation capabilities of the more powerful edge nodes to meet efficiency constraints for real-time applications. Many symmetric-key-based schemes rely on an online trusted cloud admin (CA) to establish session keys between IoT devices and edge nodes. In this study, we propose a new provably-secure mutual authentication privacy-preserving protocol with forward secrecy (MAPFS), which eliminates the requirement for an online CA during IoT authentication. To achieve anonymity, our construction utilizes zero-knowledge proofs and randomizes the IoT authentication request. The security of our construction is based on the well-studied discrete logarithm and decisional Diffie–Hellman assumptions in elliptic curve groups. We formally prove that MAPFS ensures mutual authentication and semantic security for session keys. We also evaluate MAPFS performance in terms of the communication overhead, storage requirements, and computation complexity. Finally, we test the performance of MAPFS on a Raspberry Pi 4 and compare it against other certificate-less protocols.**

***Index Terms*—Anonymity, edge computing, elliptic curve cryptography (ECC), IoT, mutual authentication (MA), zero-knowledge proof (ZKP).**

I. Introduction

CONVENTIONAL cloud computing paradigm suffers from a single point of failure. Moreover, low-end devices, henceforth referred to by IoTs cannot meet the required Quality of Service (QoS) due to the high latency in the propagation of data between the IoT device and the cloud. Researchers recommend migrating to the three-tier IoT–Edge–Cloud paradigm due to its distributed nature where IoTs can benefit from the computation of the nearby edge nodes [1], [2]. The IoT–Edge–Cloud paradigm ensures low latency, enabling IoTs to meet the required QoS besides other benefits, such as location awareness and scalability.

Mutual authentication (MA) protocols in the IoT–Edge–Cloud paradigm [3] can be categorized into symmetric key-based protocols and asymmetric key-based protocols. Symmetric key-based protocols are characterized by high efficiency and low computation complexity. Nevertheless, they require preshared key parameters between the communicating entities which is unrealistic with the enormous number of the IoTs. One approach to address this challenge is the utilization of a cloud admin (CA), which maintains a secret key for each entity where the role of the CA involves authenticating the communicating entities and deriving session keys. On the other hand, in asymmetric key-based protocol, public key encryption is not used very often for data encryption since public key encryption is costly [4]. Instead, entities public key are used to establish a session key. Then, by adopting a well-known symmetric encryption scheme, fast and secure communications become feasible. However, in certificate-based protocols [5], anonymity can be achieved through anonymous authenticated schemes, such as ring/group signature schemes [6], [7]. In such schemes, both the IoT device and the edge node should have access to public keys of registered IoT devices in the anonymity set for signature generation and verification which requires either extra communication between IoT devices or storage requirements. Moreover, registration of a new IoT device to the system requires updating all IoT devices and edge nodes with the public key of the new IoT device which further affects the system's scalability.

In order to address the aforementioned problems in anonymous authenticated schemes, we adopt a certificate-less public-key cryptography (CL-PKC) authentication and key agreement scheme where the generation of the private keys and public keys are split between the IoT device and the key generation center (KGC) [8]. Li et al. [9] proposed an elliptic curve cryptography (ECC)-based MA and key exchange protocol for IoTs. The proposed protocol achieves MA and forward secrecy property. The MA property ensures the legitimacy of the transacting entities while forward secrecy property ensures the security of the past session keys in case of the compromise of long-term secrets. However, Li's protocol failed to maintain the privacy of the IoTs since the communicating entities have to send their identities for completing the authentication process. Similarly, Ying and Nayak [10] proposed an ECC-based MA and key establishment protocol in 5G networks. The proposed protocol achieves MA and forward secrecy.

Manuscript received 20 June 2023; revised 15 August 2023; accepted 16 September 2023. Date of publication 26 September 2023; date of current version 21 February 2024. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and in part by the Fonds de Recherche du Québec Nature et Technologies (FRQNT). *(Corresponding author: Riham AlTawy.)*

Mohamed Seifelnasr is with the Department of Computer Engineering, Helwan University, Cairo 11792, Egypt, and also with the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC H3G 1M8, Canada (e-mail: m_seifel@encs.concordia.ca).

Riham AlTawy is with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8P 5C2, Canada (e-mail: raltawy@uvic.ca).

Amr Youssef is with the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC H3G 1M8, Canada (e-mail: youssef@ciise.concordia.ca).

Essam Ghadafi is with the School of Computing, Newcastle University, NE4 5TG Newcastle Upon Tyne, U.K. (e-mail: essam.ghadafi@ncl.ac.uk).

Digital Object Identifier 10.1109/JIOT.2023.3318180

Regarding anonymity, the protocol obfuscates the real identity of the IoT device. However, it suffers from linkability where an external adversary can link the authentication requests of the IoT device since it has the same pseudo-anonymous identity. Therefore, in this article, we consider the anonymity of the sender and the unlinkability of the authentication request with respect to the serving edge node.

Goals: Motivated by the aforementioned protocols, in this work, we continue this line of research and further aim to achieve the following goals.

1) Design a privacy-preserving MA protocol for the IoT–Edge–Cloud paradigm where an external adversary or edge node, controlled by a malicious network admin, cannot efficiently identify, or co-relate the incoming requests.
2) Enable scalability of the scheme without the overhead of updating all other IoT devices and edge nodes with public keys of new IoT devices.

Our Contributions: This article has the following contributions:

1) We propose MAPFS, a MA privacy-preserving protocol with forward secrecy for the IoT–Edge–Cloud. MAPFS is resilient to replay attacks; also, it achieves forward and backward secrecy properties and it ensures unlinkability between IoT requests with respect to the edge node.
2) Based on the computationally intractable EC discrete logarithm problem (ECDLP) and EC decisional Diffie–Hellman assumption (ECDDH), we formally prove the secrecy of the session key, and the MA property of the protocol.
3) We provide performance evaluations for MAPFS and compare it with other protocols in terms of the execution time.
4) We perform experiments on a 1.5 GHz 64-bit Quad-core ARM Cortex-A72 processor to validate the soundness of our proposed protocol. Moreover, we make our code for the implementation public on the GitHub repository.

In MAPFS, the IoT device sends an authentication token to a nearby edge node to prove its legitimacy. To achieve unlinkability between the authentication tokens of the same IoT device, the IoT device randomizes such authentication tokens. Such randomization process is essential as it provides the unlinkability between the authentication requests and prevents the external adversaries and the edge nodes from correlating the authentication requests and profiling the IoT device. Considering the attack scenario where an adversary builds another system (i.e., a different TTP with a different public key) that acts like our proposed system and sends a randomized request that will pass the verification on the edge node side, the IoT device provides a zero-knowledge proof (ZKP) of knowledge of the random value, without disclosing it, that relates the randomized authentication request to the public parameter of our system. In case of a misbehaving IoT device, a TTP can do a linear search on the transmitted request to get the real identity of the misbehaving IoT device.

The remainder of this article is organized as follows. Section II briefly reviews the related work and provides background on secure hash functions and the computationally intractable ECDL and ECDDH problems. Section IV presents the syntax and the security model for the system. Section V contains the details of our construction. Formal security analysis of MAPFS with respect to MA and session key secrecy is presented in Section VI. Section VII presents the performance evaluation with respect to the storage requirements, computation complexity, and communication cost. Finally, our work is concluded in Section VIII.

II. Literature Review

Numerous MA symmetric key-based protocols [11], [12], [13], [14], [15], [16] have been proposed for IoT communication. Most of these protocols utilize lightweight operations, such as XOR and hash functions. Hence, these protocols have the advantage of low-computation complexity which makes them suitable for low-end devices. However, the key distribution and management of the symmetric-key-based schemes impose a burden on practical applications of these protocols, especially with the increasing number of IoT devices.

To address the key distribution and management shortcomings in the symmetric-key schemes, asymmetric key-based protocols, such as [17] and [18] are proposed. Such protocols require only the communicating entities to exchange keying materials to establish the session keys. The applicability of such protocols in the IoT has one major inconvenience, which is the computation cost and energy consumption. Subsequently, more efforts were exerted to realize ECC-based authentication protocols that achieve the required security level with smaller parameters [19]. Li et al. [20] proposed a MA and key exchange protocol for wireless sensor networks based on ECC. Later, Shi and Gong [21] pointed out that the proposed protocol in [20] does not provide MA or forward secrecy and proposed a more secure ECC-based protocol. However, Choi et al. [22] showed that the protocol in [21] is vulnerable to session key attacks. In order to achieve a 2-factor authentication protocol, Chang and Le [23] proposed a MA scheme using a smart card, which requires a small overhead and achieves forward secrecy property. However, the proposed protocol could not resist stolen smart card attacks and tracking attacks as indicated in [24]. All the aforementioned symmetric-key-based protocols necessitate the presence of an online trusted third party during the authentication process.

CL-PKC schemes proposed in [9], [10], [25], [26], [27], [28], [29], [30], and [31] allow low-end devices to perform the authentication without the need for an online CA. Protocols in [28], [30], [31], [32], [33], and [34] are based on heavy bilinear-pairing operations, are not suitable for limited-resource low-end devices. Gayathri et al. [26] proposed an efficient certificate-less protocol that does not require bilinear-pairing operations. However, the proposed scheme does not achieve the confidentiality of the transmitted messages from the sensor nodes. More authentication protocols that do not require bilinear-pairing operations are presented in [9] and [25]. Nevertheless, the sender identity has to be sent in the clear on the wireless channel between the IoT and the edge node. The schemes proposed in [10], [27], and [29] guarantee the anonymity of the sender against an external adversary

by obfuscating its identity. However, the service provider (i.e., the receiver) can link and relate the incoming requests. The aforementioned protocols assign the low-end IoT devices to a certain serving edge node which does not fit the dynamic nature of the IoT sensors. Certificate-less schemes in [10], [28], [30], [32], and [33] achieve anonymity of the sender with respect to the serving edge node. Nonetheless, none of the certificate-less protocols mentioned above ensure session unlinkability with respect to the serving edge node.

Table I provides a comparison of the security properties of our proposed protocol, MAPFS, against other related ones. In our comparison, we consider the MA, the anonymity of the sender and the unlinkability of the IoT requests from external adversaries and edge nodes. Additionally, we consider other functional properties, such as scalability, where adding an IoT device is easy by issuing an authentication token to the new device without requiring protocol reinitialization or reregistration of other IoT devices. Our protocol, MAPFS, achieves MA, sender anonymity and session unlinkability. Note that PUF-based protocols [9], [15], [16] require the presence of a PUF circuit for running the protocol. If the PUF circuit is not present, the protocol cannot be executed, and it will lose its hardware compromise resilience security property.

III. Preliminary

Throughout our work, we utilize the ECC system, which is a modern family of public-key cryptosystems based on the algebraic structures of the elliptic curves over finite fields. ECC security is based on the assumed difficulty of the ECDLP [19], [36]. ECC-based schemes are characterized by smaller key sizes, low arithmetic requirements, and shorter operand lengths compared to RSA systems. Let p be a prime number and let $\mathbb{F}_p$ denote the field of the integers modulo p. An elliptic curve E is defined over the finite field $\mathbb{F}_p$ by the set of the points $(x,\ y) \in\ \mathbb{F}_p \times \mathbb{F}_p$ satisfying non singular elliptic curve equation $y^2 = x^3 + ax + b \bmod p$ such that $4a^3 + 27b^2 \neq 0 \bmod p$ plus the point at infinity $\mathcal{O}$. Then, the additive elliptic curve cyclic group $\mathcal{G}$ is defined as $\mathcal{G} = \{(x, y) : x, y \in\ \mathbb{F}_p \wedge (x, y) \in E\} \bigcup \mathcal{O}$. Let $P \in E$ is the generator point of the group $\mathcal{G}$ of order q. Note that, we use lowercase letters for representing scalar values in Z_q and uppercase letters for EC points in the group $\mathcal{G}$. The ECDLP problem is defined as follows.

Definition 1 (ECDLP): Let E be an elliptic curve over the finite field $\mathbb{F}_p$ and let the points R and S be points in $\mathcal{G}$ of order q, the ECDLP is the problem of finding an integer r such that $R = rS$.

The Elliptic Curve Diffie–Hellman (ECDH) [37]: ECDH is a key agreement protocol that enables two entities, Alice and Bob, having an elliptic curve public-private key pair, to establish a shared secret key over an insecure channel. ECDH is based on the ECDLP instead of the conventional discrete log problem (DLP). It works as follows. The two communicating entities, Alice and Bob agree on an elliptic curve E. Alice selects an integer $a \leftarrow Z_q^*$, computes $Q = aP$ and sends Q to Bob. On the other hand, Bob selects an integer $b \leftarrow Z_q^*$, computes $Q = bP$, computes $R = bP$, and sends R to Alice. Alice and Bob receives R and Q, respectively, and computes the shared secret key S; $S = aR = bQ = abP$. Both entities, Alice and Bob get the same value for S, and the shared key is established.

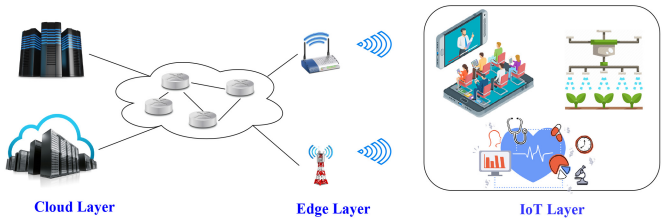

Fig. 1. IoT–Edge–Cloud paradigm.

Definition 2 (Elliptic-Curve Decisional Diffie–Hellman Problem (ECDDHP) [19]): Let E be an elliptic curve over the finite field $\mathbb{F}_p$. Given the point $P \in \mathcal{G}$ with order q and the points $X = xP$, $Y = yP$, and $Z = zP \in \mathcal{G}$, the ECDDHP is the problem of determining whether $Z = xyP$, equivalently, whether $z = xy \bmod q$.

Definition 3 (Negligible Function [38]): Given a security parameter λ, a function $\epsilon(\lambda) : \mathbb{Z} \rightarrow \mathbb{R}$ is said to be negligible in λ if for all $d > 0$, there exists $\lambda_d \in \mathbb{Z}$ such that $\epsilon(\lambda) \leq 1/\lambda^d$ for all $\lambda > \lambda_d$.

Schnorr ZKP [39]: Given a statement $\mathcal{X}$ where $\exists\ x$ s.t. $X = xP$, a Schnorr ZKP of Knowledge enables the prover to convince the verifier of the knowledge of the witness x without revealing its value to the verifier. Schnorr protocol is a Σ protocol that consists of three interactions between the prover and verifier. These interactions are: 1) commit; 2) challenge; and 3) response. A noninteractive Schnorr ZKP in the Fiat–Shamir Heuristic transformation [40] allows the prover to combine the commit, challenge, and response phases in one interaction. This transformation involves using a secure cryptographic hash function to issue the challenge.

Schnorr Signature [41]: A noninteractive Schnorr signature over the message m runs as follows. The prover generates the commitment $R = rP$ where $r \leftarrow Z_q$ and uses the Fiat–Shamir Heuristic transformation for computing the challenge $c = H(R, m)$. Then, it computes the response $s = r + cx$ where the signature $\sigma = (R, s)$. The verifier checks if $sP \stackrel{?}{=} R + cX$ hold.

IV. System Model and Threat Model

In what follows, we present our system model and threat model. Additionally, we provide a list of abbreviations used throughout the paper in Table I.

A. System Model

We adopt the IoT–Edge–Cloud paradigm as illustrated in Fig. 1. Our system model is composed of the IoT layer, the edge layer, and the cloud layer. Edge nodes are connected to the cloud layer through the core network. The role of each layer is detailed as follows.

1) *IoT Layer:* Such layer is composed of the sensor/IoT devices, which are limited-resource devices with Internet connectivity and enrolled in many applications (e.g., transportation, healthcare, virtual reality, . . . , etc.). IoT devices sense and gather the needed information and

TABLE I
COMPARISON WITH OTHER RELATED PROTOCOLS

Security/Functionality Properties	[35]	[9]	[10]	[29]	[26]	[27]	[28]	[25]	[15]	[16]	[32]	[33]	[30]	[31]	[34]	Ours
Mutual Authentication	×	✓	✓	✓	×	✓	×	✓	✓	✓	✓	✓	✓	✓	×	✓
Anonymity w.r.t External adversary	×	×	✓	✓	×	✓	×	×	✓	✓	✓	✓	✓	✓	×	✓
Anonymity w.r.t edge node	×	×	✓	×	×	×	×	×	✓	✓	×	✓	✓	✓	×	✓
Unlinkability w.r.t External adversary	×	×	×	✓	×	✓	×	×	✓	✓	✓	✓	✓	✓	×	✓
Unlinkability w.r.t edge node	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	✓
Bilinear-Pairing Free	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	×	×	×	×	×	✓
Extra Hardware-Free	✓	×	✓	✓	✓	✓	×	×	×	×	✓	✓	✓	✓	✓	✓
Scalability	✓	✓	✓	×	×	×	✓	✓	×	×	✓	✓	✓	✓	✓	✓
IoT-Edge-Cloud compatibility	×	✓	✓	×	✓	×	✓	✓	✓	✓	✓	✓	✓	×	✓	✓

send most of the data to the edge layer to benefit from the computation power of the IoT gateway in order to meet the expected QoS requirements.

2) *Edge Layer:* An intermediate layer between the IoT and cloud layers. It brings a part of the cloud computing infrastructure closer to the end-users in the form of IoT gateways to improve the latency of real-time applications. The computation-capable edge nodes can be a router, a switch, or a proxy server [42]. It offers computation offloading services to limited-resource IoT devices.
3) *Cloud Layer:* A top layer that provides scalable computing resources, storage, and services that complement edge computing. It enables data storage, complex data analytics, machine learning, and other resource-intensive tasks that cannot be efficiently performed at the edge nodes.

Moreover, in our protocol, we make use of a TTP which is the KGC that is responsible for initializing the system parameters and issuing the signing keys for IoT devices and IoT gateways, in the registration phase.

B. Threat Model

In our paper, we adopt the Canetti–Krawczyk (CK) adversary model [43]. In this model, the adversary $\mathcal{A}$ can eavesdrop, insert, modify, and drop messages on the communicating channel. Moreover, it has access to the memory of the communicating entities (i.e., IoT devices and IoT gateways). Therefore, the stored information in the memory of the IoT device and the IoT gateway is vulnerable to memory leakage attacks. Based on the information revealed to $\mathcal{A}$, we define the adversary attacks as follows.

1) *Session State Reveal:* $\mathcal{A}$ gets access to the ephemeral secrets for the current session.
2) *Session Key Query:* $\mathcal{A}$ gets access to the current session key of the communicating entities.
3) *Party Corruption:* $\mathcal{A}$ has access to the long-term keys of the communicating entity.

V. PROTOCOL DESIGN

Our protocol, MAPFS, makes use of two-party ECDH key exchange [37], Schnorr ZKP of discrete log knowledge [39], and Schnorr signature [41]. The Diffie–Hellman protocol enables the IoT device and the IoT gateway to establish the session key. However, the ECDH is an unauthenticated key agreement protocol. Therefore, the Schnorr signature is used to authenticate the IoT gateway to the IoT device, and vice versa. The IoT gateway uses a signing key, issued by the KGC, to sign its authentication message for the IoT device. Similarly, the IoT device uses a signing key to sign its authentication message. In order to maintain the unlinkability of the IoT requests, the protocol randomizes the authentication message of the IoT device by multiplying it with a random number. Furthermore, the Schnorr ZKP is used to prove the knowledge of the random number that relates the randomized authentication token to our public system parameters.

TABLE II
NOTATION TABLE

Notation	Description
QoS	Quality-of-Service
PKI	Public-Key Infrastructure
CA	Cloud Admin
ECC	Elliptic Curve Cryptography
DLP	Discrete Log Problem
ECDLP	Elliptic Curve Discrete Log Problem
ECDDH	Elliptic Curve Decisional Diffie-Hellman
ZKP	Zero-Knowledge Proof
KGC	Key Generation Center
P	EC generator point
s_{gc}	Secret Key of the KGC
Pub_{gc}	Public key of the KGC
ID_a	An IoT device with identity a
x_a, y_a	The partial private keys of the IoT device
X_a, Y_a	The partial public keys of the IoT device
σ_a	The IoT device signing key
ID_w	A gateway with identity w
x_w, y_w	The partial private keys of the gateway
X_w, Y_w	The partial public keys of the gateway
σ_w	The gateway signing key
r_1, r_2, r_3, r_4	Random nonces generated by the IoT device
r_5	Random nonce generated by the gateway
A	A one-time public key for the IoT device
W	A one-time public key for the IoT gateway
P_1	An instantaneous base point generated by the IoT device
P_2, P_3	Two randomized points generated by the IoT device
T_1, T_2	ZKP commitments
s_1, s_2	ZKP responses
σ_t	The generated IoT signature
σ_z	The generated gateway signature
K_s	The session key

Our protocol consists of a setup phase, registration phase, MA and key agreement phase, and revocation phase. The details of each phase are listed as follows.

A. Setup Phase

In this phase, the service provider deploys IoT gateways. Then, the KGC publishes the system public parameters, i.e., the used hash functions, the utilized elliptic curve, and the

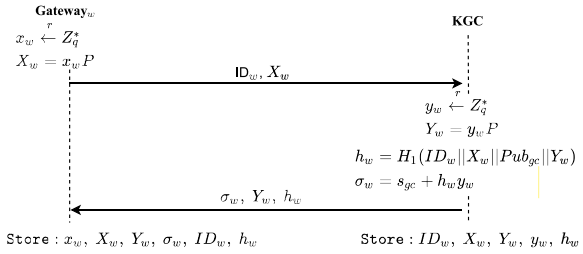

Fig. 2. IoT gateway registration.

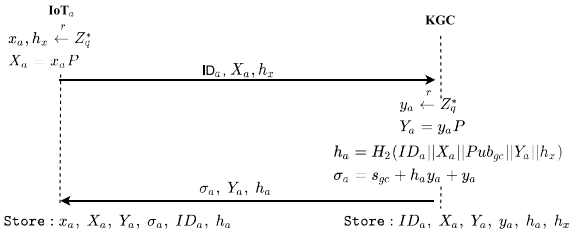

Fig. 3. IoT device registration.

KGC public key. The KGC selects the system parameters as follows. The KGC chooses a 256-bit prime number p and an elliptic curve E over the finite field $\mathbb{F}_p$. The KGC chooses a point $P \in E$ of order q as the generator point. It randomly selects $s_{gc} \xleftarrow{r} Z_q^*$ as its secret key and computes the public points $\text{Pub}_{gc} = s_{gc}P$. Also, it selects the collision-resistant one-way hash functions $H_0 : \mathcal{G} \rightarrow Z_q^*$, $H_1 : \{0,1\}^{128} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G} \rightarrow Z_q^*$, $H_2 : \{0,1\}^{128} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \{0,1\}^{256} \rightarrow Z_q^*$, $H_3 : \mathcal{G} \times \mathcal{G} \rightarrow Z_q^*$, $H_4 : \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G} \rightarrow Z_q^*$, $H_5 : \{0,1\}^{256} \rightarrow Z_q^*$. Finally the public parameters of the system (E, p, q, P, H_0, H_1, H_2, H_3, H_4, H_5, Pub_{gc}) are published while s_{gc} is kept secret.

B. Registration Phase

We assume that the registration phase runs in a secure environment. Fig. 2 illustrates the registration phase between the gateway$_w$ and the KGC. The IoT gateway randomly selects $x_w \xleftarrow{r} Z_q^*$ as its partial private key and computes the partial public key $X_w = x_w P$. Then, the IoT gateway sends its identity ID_w along with the partial public key X_w to the KGC. After that, the KGC randomly selects $y_w \xleftarrow{r} Z_q^*$ and computes the partial public key $Y_w = y_w P$ (i.e., the two partial keys X_w, Y_w are generated by the IoT gateway and the KGC, respectively, and serve as the public key of the gateway). Then, the KGC computes $h_w = H_1(ID_w||X_w||\text{Pub}_{gc}||Y_w)$ to bind the public parameters ID_w, X_w, Pub_{gc}, and Y_w, together. Then, the KGC issues the signing key $\sigma_w = s_{gc} + h_w y_w$ which is verified on the gateway by validating $\sigma_w P \stackrel{?}{=} \text{Pub}_{gc} + H_1(ID_w||X_w|||\text{Pub}_{gc}||Y_w)Y_w$. Note that, the KGC includes the secret s_{gc} to indicate issuing of the signing key by the KGC and binds the h_w value with the partial private key of the gateway y_w to indicate that this signing key is issued to the IoT gateway with public key Y_w.

Similar to the registraion of IoT gateway, the KGC includes s_{gc} and $h_a y_a$ in the IoT signing key to indicate issuing of such signing key by the KGC. Moreover, the KGC relates the h_a with the partial private key of the IoT device y_a to indicate that this signing key is issued to the IoT device with public key Y_a as shown in Fig. 3. However, since our protocol is privacy-preserving and the IoT authentication request should be unlinkable, the registration phase of the IoT device is a little bit different from the gateway registration. The KGC computes the $h_a = H_2(ID_a||X_a|||\text{Pub}_{gc}||Y_a||h_x)$ to bind the IoT identity ID_a, the IoT public key $<X_a, Y_a>$, and the KGC public key Pub_{gc}, together. Moreover, the KGC includes the secret key h_x in the computation of h_a to prevent the exhaustive searching by an external adversary and identifying the IoT device identity through hashing the public parameters ID_a, $<X_a, Y_a>$, Pub_{gc}, in the IoT authentication request, if h_x is not included. Furthermore, the KGC includes the term y_a in the signing key of the anonymous IoT device in order to provide a ZKP of the knowledge of the discrete log, h_a, of the randomized points P_3 to the base point P_1. Without a valid value of h_a that links $P_3 = h_a P_1$, the KGC cannot revoke the identity of a misbehaving IoT device.

To enhance the scalability and alleviate the bottleneck problem during the registration of IoT devices, we allowed an IoT device to send its registration data $<ID_a||X_a||h_x>$ and a symmetric key s_k to the KGC through deployed gateways (i.e., note that, in this case, the public key of the KGC Pub_{gc} is assumed to be embedded on the IoT device). Afterward, the KGC decrypts the incoming data and generates the partial public key Y_a for the IoT device along with the signing key σ_a and the $h_a = H_2(ID_a||X_a|||\text{Pub}_{gc}||Y_a||h_x)$. Then, it does symmetric encryption using the secret s_k for $<ID_a||X_a||\sigma_a||Y_a||h_a>$ and sends the encrypted data to the IoT device through edge nodes along with the integrity term $M_1 = H(ID_a||X_a||\sigma_a||Y_a||h_a)$. The IoT device decrypts the incoming message and verifies $M_1 \stackrel{?}{=} H(ID_a||X_a||\sigma_a||Y_a||h_a)$ and $\sigma_a P \stackrel{?}{=} \text{Pub}_{gc} + h_a Y_a + Y_a$. Then, the IoT device stores σ_a, x_a, and h_a in its memory. This solution enhances the scalability and alleviates the bottleneck problem during the registration of IoT devices but it requires the public key of the KGC to be embedded in the IoT device. Another alternative to enhance the scalability and alleviate the bottleneck problem is to utilize a hierarchical structure comprising a root KGC and sublocal KGCs [44]. This architecture enables IoT devices to register with their respective sublocal KGCs, providing a more distributed and efficient approach.

C. Mutual Authentication and Key Agreement Phase

Consider a session i between IoT$_a$ and gateway$_w$, as shown in Fig. 4, the IoT$_a$ randomly generates the random nonces r_1, r_2, r_3, and r_4 where r_1 is used in deriving the IoT one-time public key $A = r_1 X_a$, r_2 is used in randomizing the IoT signing key, σ_a while r_3 and r_4 are used in generating the ZKP commitments T_1 and T_2. The IoT sends "Hello" message along with the IoT one-time public-key A to the IoT gateway. Then, the IoT gateway generates r_5 and computes the one-time public key $W = r_5 X_w$. Afterwards, it computes I_g and the gateway

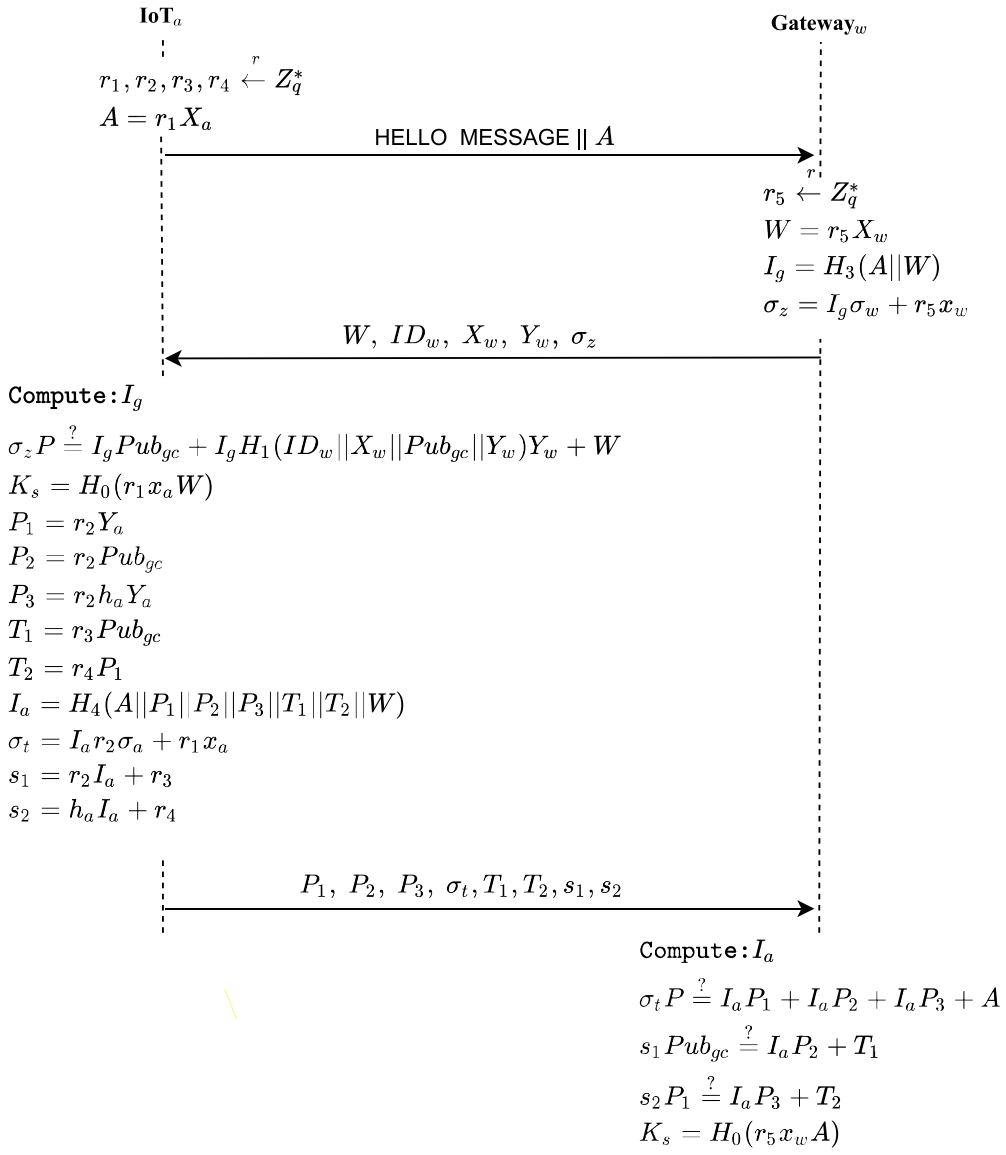

Fig. 4. Mutual authentication phase.

signature $\sigma_z = I_g \sigma_w + r_5 x_w$ such that $I_g = H_3(A||W)$. Such σ_z will be verified by the IoT device by checking $\sigma_z P \stackrel{?}{=} I_g \text{Pub}_{gc} + I_g H_1(ID_w||X_w||\text{Pub}_{gc}||Y_w) Y_w + W$. Upon receiving the IoT gateway message, $<W, ID_w, X_w, Y_w, \sigma_z>$, the IoT device verifies the received σ_z. Upon the successful verification of the received σ_z and authenticating the IoT gateway, the IoT device computes the session key $K_s = H_0(r_1 x_a W)$; otherwise, it terminates the session with such IoT gateway. After that, it computes the instantaneous base point $P_1 = r_2 Y_a$, and randomized points $P_2 = r_2 \text{Pub}_{gc}$ and $P_3 = r_2 h_a Y_a$. For the ZKP, the IoT device computes the two commitments $T_1 = r_3 \text{Pub}_{gc}$ and $T_2 = r_4 P_1$ to prove the knowledge of r_2 (resp. h_a) in the statement $\exists\ r_2$ s.t. $P_2 = r_2 \text{Pub}_{gc}$ (resp. $\exists\ h_a$ s.t. $P_3 = h_a P_1$). After that, the IoT device computes $I_a = H_4(A||P_1||P_2||P_3||T_1||T_2||W)$. Then, the IoT computes the randomized signature $\sigma_t = I_a r_2 \sigma_a + r_1 x_a$ that will be verified by the IoT gateway by checking $\sigma_t P \stackrel{?}{=} I_a(P_1 + P_2 + P_3) + A$. Also, the IoT computes the ZKP responses $s_1 = r_2 I_a + r_3$, $s_2 = h_a I_a + r_4$. Later on, the IoT gateway verifies these ZKP responses by checking $s_1 \text{Pub}_{gc} \stackrel{?}{=} I_a P_2 + T_1$ and $s_2 P_1 \stackrel{?}{=} I_a P_3 + T_2$. The IoT device sends points P_1, P_2, P_3 along with the commitments T_1, T_2, IoT signature σ_t and ZKP response s_1, s_2 to the IoT gateway. Then, the IoT gateway verifies the IoT signature σ_t and the ZKP responses s_1 and s_2 for authenticating the IoT device.

D. Revoking the Anonymity of Misbehaving IoTs

Since the protocol is privacy-preserving and the IoT gateway cannot identify the sender IoT, it is required that the KGC can retrieve the identity of an IoT request in case of misbehavior. The KGC achieves this by performing a linear search and computing $P_3 = h_j P_1$ where $1 \leq j \leq n$ and n is the total

number of the registered IoTs. Upon successful passing of the check, the identity of the IoT device is determined as ID_j.

Then, the KGC sends $<ID_j, y_j, X_j, Y_j, h_j>$ to the deployed IoT gateways. An IoT gateway stops serving the IoT device with ID_j by checking $P_3 = h_j P_1$. If it holds, the IoT device is banned from the computation offloading service.

VI. Security Analysis

In this section, we analyze the security properties of MAPFS. We start by modeling the adversarial capabilities. Then, we prove the semantic security (SS) of our key exchange protocol and its MA property. Furthermore, we show MAPFS resilience to replay attacks and how MAPFS maintains the perfect forward secrecy and backward secrecy properties.

A. Adversary Model

We adopt the CK-threat model where a PPT $\mathcal{A}$ has access to the public wireless communication channel and can eavesdrop, modify, and inject messages. Moreover, in this model, $\mathcal{A}$ can get access to the information in the participant's memory, such as the long-term secrets (i.e., x_a, σ_a, h_x, h_a in case of an IoT device and x_w, σ_w in case of an IoT gateway) or the internal state variables (i.e., r_1, r_2, r_3, and r_4 in case of an IoT device and r_5 in case of an IoT gateway) used in deriving the session key. Therefore, the leakage of these secrets inside the IoT device memory should have the least effect on the security of the protocol.

In our proposed protocol, we define the session identifier $\mathcal{S} = H_3(A||W)$ as the hash value of the IoT device one-time public key A and the gateway one-time public key W. Moreover, the protocol participants, IoT_a and IoT gateway_w, are said to be partners if the following conditions are met: 1) the two participants are in the accept state; 2) they have the same session identifier $\mathcal{S}_i = <A||W>$; and 3) the partner identifier of IoT_a is IoT gateway_w and vice versa [45]. We model the adversary capabilities in interacting with the protocol participant $\mathcal{O}$ (i.e., IoT device or the IoT gateway) by the following queries.

1) *H( ):* The hash function is simulated as a random oracle. For each simulation H_i, a list $\mathcal{L}_i$ is maintained to keep the input in_i and the output out_i. When queried by $\mathcal{A}$, if the input in_i is found in the stored list $\mathcal{L}_i$, the output out_i is returned; otherwise, a random string $\text{out}_i \xleftarrow{r} \{0, 1\}^{l_i}$ is returned where l_i is the output length of the hash function H_i and the entry in_i and out_i is added to the stored list $\mathcal{L}_i$.
2) *Send*$(\mathcal{S}, m, \mathcal{O})$*:* It is used to model the adversary's active attacks on the system, such as replay attacks, impersonation attacks, and injection attacks. It allows the adversary to act as a legitimate entity and send a message m to the protocol participant $\mathcal{O}$ in the session $\mathcal{S}$. It responds according to the protocol specifications, which depend on its role and current internal state.
3) *Execute*(IoT_a, G_w)*:* It is used to model the adversary's passive attacks by generating the transcript $\mathcal{T}$ of the transmitted messages during an honest protocol execution between the protocol participants, namely, the IoT device IoT_a and the IoT gateway G_w.
4) *SSReveal*$(\mathcal{O}, \mathcal{S})$*:* This query allows $\mathcal{A}$ to obtain the internal state of the protocol participant $\mathcal{O}$ during the execution of the protocol in session $\mathcal{S}$, including any relevant internal variables such as r_1, r_2, r_3, and r_4 for the IoT device and r_5 for the IoT gateway.
5) *SKReveal*$(\mathcal{O}, \mathcal{S})$*:* This query allows $\mathcal{A}$ to determine the session key (i.e., K_s in our protocol) held by the participant $\mathcal{O}$ during the session $\mathcal{S}$.
6) *Corrupt*$(\mathcal{O})$*:* This query allows $\mathcal{A}$ to obtain the long-term secrets used by the participant $\mathcal{O}$ in the protocol, such as x_a, σ_a, and h_a for the IoT device and x_w and σ_w for the IoT gateway.

MA: Intuitively, we say that MAPFS ensures MA, if it is infeasible for $\mathcal{A}$ to impersonate an IoT device to an honest gateway, and it is also infeasible for $\mathcal{A}$ to impersonate a gateway to an honest IoT device. The MA security of a MAPFS scheme supporting n IoT devices and m gateways is modeled by an experiment Auth, where $\mathcal{A}$ attempting to impersonate IoT_j (resp. gateway G_j) is allowed to invoke the **Send** query with $\{\text{IoT}_1, \text{IoT}_2, \ldots, \text{IoT}_n\} - \{\text{IoT}_a\}$ (resp. $\{G_1, G_2, \ldots, G_m\} - \{G_j\}$). At the end, $\mathcal{A}$ wins if it outputs a valid login message for the target IoT device IoT_a or the target IoT gateway G_w. The advantage of $\mathcal{A}$ is denoted by $\text{Adv}^{\text{Auth}}_{\text{MAPFS}}(\mathcal{A})$.

The session key K_s of $\mathcal{O}$ is said to be fresh if the following conditions hold: 1) no **SKReveal** has been invoked on $\mathcal{O}$ or its partner and 2) at most one corrupt query, either **SSReveal** or **Corrupt** has been invoked by $\mathcal{A}$ on $\mathcal{O}$ or its partner. It is reasonable that if $\mathcal{A}$ gets the secret parameters of both the protocol participants, $\mathcal{A}$ can compute the session key [9], [45].

SS: The SS of MAPFS is violated if $\mathcal{A}$ distinguishes a fresh session key K_s from a random sequence. The SS of MAPFS is modeled by an indistinguishability experiment *SSec* where $\mathcal{A}$ repeatedly invokes **Execute**, **SSReveal**, **Corrupt**, **SKReveal** on some protocol participants for n_q times. Finally, in the challenge phase, an unbiased coin c is flipped. If the flipped coin $c = 1$, a fresh session key K_s for a valid protocol transcript $\mathcal{T}$ of MAPFS partners is output to $\mathcal{A}$; otherwise, a random key of the same length is output. $\mathcal{A}$ responds by outputting a bit c'. $\mathcal{A}$ wins the experiment if $c' = c$. The advantage of $\mathcal{A}$ is denoted by $Adv^{SSec}_{\text{MAPFS}}(\mathcal{A})$.

B. Security Analysis

Using the security model discussed before, in what follows, we prove that MAPFS achieves both MA and SS in the random oracle model.

Theorem 1: Under the assumption of the intractability of ECDLP, MAPFS is MA-secure where for any PPT adversary $\mathcal{A}$, $\text{Adv}^{\text{Auth}}_{\text{MAPFS}}(\mathcal{A}) \leq \epsilon$.

Proof: We proceed by showing that for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in impersonating an IoT device is negligible. We show that if $\mathcal{A}$ can break the IoT-to-Gateway authentication, it can be used as a subroutine in adversary $\mathcal{B}$ who can break the ECDLP. Given the EC point $Q = sP$ such that $s \xleftarrow{r} Z^*_q$, $\mathcal{B}$ simulates the protocol and solves the ECDLP as follows.

In the initialization phase, $\mathcal{B}$ initializes the protocol and sets the public key $\text{Pub}_{gc} = Q$. Also, it sets the IoT device IoT_a as the target device for $\mathcal{A}$ (i.e., it is required to impersonate the IoT device IoT_a and generate a valid login message). $\mathcal{B}$ generates $r, y_a \xleftarrow{r} Z_q^*$ and computes $Y_a = y_a P, A = rP, h_a = H_2(ID_a||X_a||\text{Pub}_{gc}||Y_a||h_x)$, where ID_a and X_a are the normal parameters for IoT_a in our protocol. Also, $\mathcal{B}$ initializes the lists $\mathcal{L}_{\text{IoT}} = \{\}, \mathcal{L}_2 = \{\}$ and stores h_x, y_a in $\mathcal{L}_{\text{IoT}}$ and h_a in $\mathcal{L}_2$. It also performs the **Send** query to send <"Hello", A > on the protocol participant, IoT gateway w, to get the tuple $<W, ID_w, X_w, Y_w, \sigma_z>$.

In the training phase, $\mathcal{A}$ performs the **Send** query to send the tuple $<W, ID_w, X_w, Y_w>$ to the protocol participants $\mathbb{S}_I = \{\text{IoT}_1, IoT_2, \cdots, IoT_n\}$ to obtain the IoTs authentication tuple $<P_1, P_2, P_3, \sigma_t, T_1, T_2, s_1, s_2>$. Note that, the target IoT device $\text{IoT}_a \notin \mathbb{S}_I$.

After the training phase, $\mathcal{B}$ notifies $\mathcal{A}$ to send a valid login message for the target IoT device IoT_a. Suppose $\mathcal{A}$ successfully submits a valid login message $<P_1, P_2, P_3, \sigma_t, T_1, T_2, s_1, s_2>$ for the target IoT device IoT_a. Then, the following conditions hold:

$$\begin{aligned} \sigma_t &= I_a r_2 \left(s_{gc} + y_a h_a + y_a\right) + r_1 x_a \\ s_1 &= r_2 I_a + r_3 \\ s_2 &= h_a I_a + r_4 \end{aligned} \tag{1}$$

where $I_a = H_4(A||P_1||P_2||P_3||T_1||T_2||W)$. According to the forking lemma [46], [47], $\mathcal{B}$ and $\mathcal{A}$ can repeat the above game with the same random nonces r_1, r_2, r_3, and r_4 and a different hash oracles $\mathcal{H}$ until $\mathcal{A}$ outputs another valid login message $<P_1, P_2, P_3, \sigma_t', T_1, T_2, s_1', s_2'>$ such that

$$\begin{aligned} \sigma_t{}' &= I_a{}' r_2 \left(s_{gc} + y_a h_a + y_a\right) + r_1 x_a \\ s_1{}' &= r_2 I_a{}' + r_3 \\ s_2{}' &= h_a I_a{}' + r_4 \end{aligned} \tag{2}$$

from (1) and (2)

$$\begin{aligned} r_2 &= \left(I_a - I_a{}'\right)^{-1}\left(s_1 - s_1'\right) \\ h_a &= \left(I_a - I_a{}'\right)^{-1}\left(s_2 - s_2'\right) \\ \left(I_a - I_a{}'\right)^{-1}\left(\sigma_t - \sigma_t{}'\right) &= r_2 s_{gc} + r_2 h_a y_a + r_2 y_a. \end{aligned} \tag{3}$$

$\mathcal{B}$ gets y_a for the target IoT device IoT_a from the list $\mathcal{L}_{\text{IoT}}$ and responds with $s = r_2{}^{-1}((I_a - I_a{}')^{-1}(\sigma_t - \sigma_t{}') - r_2 h_a y_a - r_2 y_a)$ as a solution for the ECDLP for the given point Q. However, under the assumption of the intractability of the ECDLP, $\mathcal{B}$ does not exist and accordingly, $\mathcal{A}$ who can produce a valid login message to the IoT gateway cannot exist.

Furthermore, we show that the advantage of $\mathcal{A}$ in impersonating an IoT gateway is negligible. We show that if $\mathcal{A}$ can break the Gateway-to-IoT authentication, it can be used by $\mathcal{B}$ who can break the ECDLP. Given the EC point $Q = sP$ such that $s \xleftarrow{r} Z_q^*$, $\mathcal{B}$ simulates the protocol and solves the ECDLP as follows.

In the initialization phase, $\mathcal{B}$ initializes the protocol and sets the public key $\text{Pub}_{gc} = Q$. Also, it sets the IoT gateway w as the target device for $\mathcal{A}$ (i.e., it is required to impersonate the IoT gateway w and generate a valid login message). Also, $\mathcal{B}$ generates $r, y_w \xleftarrow{r} Z_q^*$ and computes $Y_w = y_w P, A = rP, h_w = H_1(ID_w||X_w||\text{Pub}_{gc}||Y_w)$ where ID_w and X_w are the normal parameters for the gateway w in our protocol. Then, $\mathcal{B}$ initializes the lists $\mathcal{L}_w = \{y_w\}$ and $\mathcal{L}_1 = \{h_w\}$.

In the training phase, $\mathcal{A}$ performs the **Send** query to send <"*Hello*", A> to the protocol participants $\mathbb{S}_G = \{G_1, G_2, \ldots, G_m\}$ to obtain the gateway authentication tuple $<W, ID_w, X_w, Y_w, \sigma_z>$. Note that, the target IoT gateway $G_w \notin \mathbb{S}_G$.

After the training phase, $\mathcal{B}$ notifies $\mathcal{A}$ to send a valid login message for the target IoT gateway G_w. Suppose $\mathcal{A}$ successfully submits a valid login message $<W, ID_w, X_w, Y_w, \sigma_z>$ to impersonate the target IoT gateway w. Then, the following condition holds:

$$\sigma_z = I_g\left(s_{gc} + y_w h_w\right) + r_5 x_w \tag{4}$$

where $I_g = H_3(A||W)$.

According to the forking lemma, $\mathcal{B}$ and $\mathcal{A}$ can repeat the above game with the same randomness r_5 and a different hash oracle $\mathcal{H}$ until $\mathcal{A}$ outputs another valid login message $<W, ID_w, X_w, Y_w, \sigma_z>$ such that

$$\sigma_z{}' = I_g'\left(s_{gc} + y_w h_w\right) + r_5 x_w \tag{5}$$

from (4) and (5)

$$\left(I_g - I_g{}'\right)^{-1}\left(\sigma_z - \sigma_z{}'\right) = s_{gc} + h_w y_w.$$

Then, $\mathcal{B}$ gets the y_w for the target IoT gateway from the list $\mathcal{L}_w$ and responds with $s = ((I_g - I_g{}')^{-1}(\sigma_z - \sigma_z{}') - h_w y_w)$ as a solution for the ECDLP for the given point Q. However, under the hardness assumption of the ECDLP, $\mathcal{B}$ does not exist and accordingly, an $\mathcal{A}$ who can produce a valid login message to the IoT gateway cannot exist.

It follows that the probability of $\mathcal{A}$ in violating IoT-to-Gateway authentication is negligible and the probability of $\mathcal{A}$ in violating Gateway-to-IoT authentication is negligible. Thus, the advantage of $\mathcal{A}$ in violating the MA property of the protocol is negligible, and MAPFS is MA-secure. ■

Theorem 2: Under the intractability assumption of ECDDH, MAPFS is SS-secure where for any PPT adversary $\mathcal{A}$, $Adv_{\text{MAPFS}}^{SSec}(\mathcal{A}) \leq \epsilon$.

Proof: Let us assume that a PPT adversary $\mathcal{A}$ can guess the bit involved in *SSec*, then we show that there exists an adversary $\mathcal{B}$ who can solve the ECDDH problem as in the following game between $\mathcal{A}$ and $\mathcal{B}$.

Given the points $X = xP$, $Y = yP$, and Z such that $Z = xyP$ if $b = 1$ and $Z = zP$, otherwise, where x, y, and $z \xleftarrow{r} Z_q^*$.

In the initialization phase, $\mathcal{B}$ sets Pub_{gc} as the public key of the CA and sets the IoT device IoT_a and the IoT gateway G_w for protocol interaction.

In the training phase, $\mathcal{A}$ repeats the following for n_q times:

1) $\mathcal{A}$ performs **Execute** query to get the protocol transcript $\mathcal{T}_i$ between the IoT device IoT_a and the IoT gateway G_w for protocol instances π_i with session identifier $\mathcal{S}_i$, where i is the iteration number and $i = 1, 2, \ldots, n_r$ where n_r is the total number of the iterations.

2) $\mathcal{A}$ invokes **Corrupt** on either the IoT device IoT_a or the IoT gateway G_w. Moreover, $\mathcal{A}$ invokes **SSReveal** on the protocol instance π_i.
3) $\mathcal{A}$ invokes the **SKReveal** query on the protocol instance π_i to get the computed session key K_s.

In the challenge phase, $\mathcal{B}$ simulates the protocol with $\mathcal{A}$ to produce a valid protocol transcript $\mathcal{T}$ as follows. $\mathcal{B}$ randomly generates $\sigma_1 \xleftarrow{r} Z_q^*$, $I_a \xleftarrow{r} Z_q^*$, and $h_a \xleftarrow{r} Z_q^*$ and the random nonces r_1, r_2, r_3, and $r_4 \xleftarrow{r} Z_q^*$. Then, it computes $Y_a = (I_a + h_a I_a)^{-1}((\sigma_1 P - I_a \text{Pub}_{gc} - r_1 r_2^{-1} X)$. After that, $\mathcal{B}$ computes the points $A = r_1 X$, $P_1 = r_2 Y_a$, $P_2 = r_2 \text{Pub}_{gc}$, $P_3 = r_2 h_a Y_a$, $T_1 = r_3 \text{Pub}_{gc}$, and $T_2 = r_4 P_1$ and the responses $s_1 = r_2 I_a + r_3 \bmod q$ and $s_2 = h_a I_a + r_4 \bmod q$. Then, it outputs $<$"*Hello*", $A>$ as the first message $\mathcal{M}_1$ of the protocol transcript $\mathcal{T}$. To generate the second message $\mathcal{M}_2$ from the IoT gateway to the IoT device in the protocol transcript $\mathcal{T}$, $\mathcal{B}$ randomly generates $\sigma_2 \xleftarrow{r} Z_q^*$, $I_g \xleftarrow{r} Z_q^*$, $h_w \xleftarrow{r} Z_q^*$, and $r_5 \xleftarrow{r} Z_q^*$. Then, it computes $Y_w = (I_g h_w)^{-1}(\sigma_2 P - I_g \text{Pub}_{gc} - r_5 Y)$. After that, $\mathcal{B}$ produces the points $W = r_5 Y$ and adds $<A, W>$ to $\mathcal{L}_3$ as input to the hash query with output I_g and adds, also, $<ID_w,\ X_w,\ \text{Pub}_{gc},\ Y_w>$ to $\mathcal{L}_2$ as input to the hash query with output h_w. Then, it outputs $<W,\ ID_w,\ X_w,\ Y_w,\ \sigma_z>$ as the second message $\mathcal{M}_2$ from the IoT gateway to the IoT device in the protocol transcript $\mathcal{T}$. After that, the IoT device adds $<A, P_1, P_2, P_3, T_1, T_2, W>$ to $\mathcal{L}_4$ as input to the hash query with output I_a. Then, it outputs $<P_1, P_2, P_3, \sigma_t, T_1, T_2, s_1, s_2>$ as the third message $\mathcal{M}_3$ from the IoT device to the IoT gateway in the protocol transcript $\mathcal{T}$.

Next, $\mathcal{B}$ outputs the transcript $\mathcal{T}$ and the string $H_0(r_1 r_5 R)$ to $\mathcal{A}$ where $\mathcal{T}$ is indistinguishable from the transcripts produced in the training phase. Specifically, $\mathcal{A}$ validates $\mathcal{M}_2$ of the output transcript $\mathcal{T}$ by checking $\sigma_z P \stackrel{?}{=} I_g \text{Pub}_{gc} + I_g h_w Y_w + W$ where $\mathcal{A}$ checks list $\mathcal{L}_3$ for the entry $<A, W>$ to get I_g and check the $\mathcal{L}_1$ for the entry $<ID_w,\ X_w,\ \text{Pub}_{gc},\ Y_w>$ to get h_w. Also, $\mathcal{A}$ validates the third message $\mathcal{M}_3$ of the output transcript $\mathcal{T}$ by checking $\sigma_t P \stackrel{?}{=} I_a(P_1 + P_2 + P_3) + A$, $s_1 \text{Pub}_{gc} \stackrel{?}{=} I_a P_2 + T_1$ and $s_2 P_1 \stackrel{?}{=} I_a P_3 + T_2$, where $\mathcal{A}$ checks $\mathcal{L}_4$ for the entry $<A, P_1, P_2, P_3, T_1, T_2, W>$ to get I_a.

Assuming a PPT adversary $\mathcal{A}$ who can break the SS of the proposed protocol and outputs $c' = 1$ if the string $H_0(r_1 r_5 R)$ is the session key and $c' = 0$, otherwise. $\mathcal{B}$ gets the bit c' from $\mathcal{A}$ and passes it as his guess bit b' and wins the ECDDH game. Under the hardness ECDDH, there is no adversary $\mathcal{B}$ who can win the ECDDH with a nonnegligible probability, therefore, there is no such an adversary $\mathcal{A}$ who can break the SS of MAPFS. ■

C. MAPFS Freshness, Anonymity, Backward Secrecy, and Forward Secrecy Properties

1) MAPFS Freshness: MAPFS would be vulnerable to replay attacks if an adversary $\mathcal{A}$ can use an old generated IoT signature σ_t or gateway signature σ_z to impersonate either the IoT device or the IoT gateway.

In our protocol, the IoT device starts the session with "Hello" message along with a fresh one-time IoT public key $A = r_1 X_a$. The IoT gateway replies with a fresh one-time gateway public key $W = r_5 X_w$. These fresh A, W are used in the computation of the integrity terms $I_g = H_3(A||W)$ and $I_a = H_4(A||P_1||P_2||P_3||T_1||T_2||W)$ which are embedded in the gateway signature $\sigma_z = I_g \sigma_w + r_5 x_w$ and the IoT signature $\sigma_t = I_a r_2 \sigma_a + r_1 x_a$.

Therefore, a replay attack, in a session $\mathcal{S}'$ will not be valid since $\mathcal{A}$ has to include the fresh A', W' generated by the IoT device and the gateway, during the new session $\mathcal{S}'$, in the IoT signature σ_t' and the gateway signature σ_z' such that $\sigma_t' P = I_a'(P_1 + P_2 + P_3) + A'$ and $\sigma_z' P = I_g' \text{Pub}_{gc} + I_g' H_1(ID_w||X_w||Pub_{gc}||Y_w) Y_w + W'$. This replay attack, using the old σ_t and σ_z, is not valid under the preimage resistance property of the hash function and MAPFS is secure against replay attacks. This returns for the fact that the new σ_t' and σ_z' have to include the new $I_a' = H_4(A'||P_1||P_2||P_3||T_1||T_2||W')$ and $I_g' = H_3(A'||W')$ of the new A', W'.

2) Unlinkability of the IoT Requests: Upon authenticating with the IoT gateway, the IoT device sends the transcript $<A,\ P_1,\ P_2,\ P_3,\ T_1,\ T_2,\ s_1,\ s_2>$ where $A = r_1 X_a$, $P_1 = r_2 Y_a$, $P_2 = r_2 \text{Pub}_{gc}$, $P_3 = r_2 h_a Y_a$, $T_1 = r_3 \text{Pub}_{gc}$, and $T_2 = r_4 P_1$. Since all the sent parameters are randomized by r_1, r_2, r_3, and r_4, the IoT request is computationally indistinguishable from random sequence. Moreover, an exhaustive search over the registered IoT devices, to identify the requesting IoT device or correlate the IoT requests, is not applicable without knowing y_a or h_a of each IoT device which are known only to the CA. Therefore, the advantage of $\mathcal{A}$ in violating the unlinkability of the IoT requests is negligible and $\mathcal{A}$ cannot relate the IoT requests.

3) Perfect Forward Secrecy: This property is maintained if the compromise of the long-term key or the current session key does not lead to the leakage of the past session keys [48]. Here, in our protocol, the session key $K_s = H_0(r_5 x_w A) = H_0(r_1 x_a W) = H_0(r_1 r_5 x_a x_w P)$ where r_1 and r_5 are two random nonces generated by the IoT device and the IoT gateway, respectively. Therefore, the protocol is said to achieve perfect forward secrecy since the computation of the session key depends on the long-term key $x_a,\ x_w$ of the IoT device and the IoT gateway as well as the fresh randoms $r_1,\ r_5$ generated by the IoT device and the IoT gateway during the new session.

4) Backward Secrecy: This property is maintained when an adversary who has access to the protocol state values (i.e., $r_1,\ r_2,\ r_3,\ r_4,\ A,\ P_1,\ P_2,\ P_3,\ r_5,\ W$) cannot compute the previous session keys [49]. The computation of the session key of the session i depends on the randoms generated by the IoT gateway and the IoT device during the session i where $K_s = H_0(r_1 r_5 x_a x_w P)$. Therefore, compromising the IoT device state value during session i does not leak any information about the session key of the sessions $i-1,\ i-2, \ldots,\ 2,\ 1$. Hence, MAPFS achieves the backward secrecy property.

VII. Performance Evaluation

It is important to consider the efficiency of the proposed protocol by analyzing its performance in terms of the 1) communication overhead which consists of messages exchanged between the communicating entities before the actual transfer of information, i.e., these are the messages exchanged between

TABLE III
SUMMARY OF OUR PERFORMANCE ANALYSIS

Description	Value
IoT storage	176 bytes
Gateway storage	144 bytes
IoT computation	4 RNG, 10 scalar point multiplication, 2 point addition, 4 Hash.
Gateway computation	1 RNG, 10 scalar point multiplication, 4 point addition, 3 Hash.
Communication overhead	432 bytes

TABLE IV
AVERAGE CRYPTOGRAPHIC OVERHEAD TIME USING 1000 RUNS

Cryptographic Primitives	Symbol	Execution Time (msec)
Hash function [1]	T_h	0.0507
HMAC function [2]	T_{mac}	0.061
Symmetric Enc/Dec [3]	T_s	0.22
RNG	T_{rng}	0.053
Bilinear Pairing operation [4]	T_b	12.45
EC scalar Multiplication [4]	T_{sm}	0.37
EC Point Addition	T_{pa}	0.145
Modular Exponential Operation	T_e	0.87

[1]: Based on SHA-256 with 1024 bytes as input. The representation of the EC points in the domain of the hash function is done using the x and y coordinates and modular q is performed on the output hash function to get a co-domain in Z_q.
[2]: Based on SHA-256 for the message-digest algorithm
[3]: Based on a 128-bit key AES-CBC mode.
[4]: We use the Python library bplib, which implements a bilinear pairing over a Barreto-Naehrig curve [51].

the IoT device and IoT gateway to achieve the MA and session key establishment; 2) storage requirement on the IoT device and the IoT gateway, i.e., the secrets stored on the IoT device and the IoT gateway to achieve MA and session key derivation; and 3) computation cost which involves the operations that are done by the IoT device and the IoT gateway during the authentication process and session establishment. In our performance analysis, we assume 128-bit random values and 128-bit ID. Also, we assume a 256-bit elliptic curve which typically provides nearly a 128-bit security level [50]. In order to perform the hash functions H_0, H_1, H_2, H_3, H_4 which incur EC points in their domains, we use the x and y coordinates for the representation of the EC points. Moreover, as the codomains for the hash function are in Z_q, we perform modular q operation on the output of the hash function where q is a 256-bit. The summary of our performance analysis is presented in Table III.

A. Storage Requirements

The IoT device needs to store the 256-bit private key x_a, the 128-bit identity ID_a, the 256-bit signing key σ_a, the 256-bit h_a, and the 256-bit public keys X_a and Y_a which is equivalent to a storage of 11×128 bits.

On the other side, the IoT gateway needs a 9×128 storage space to keep the 128-bit private key x_w, the 256-bit signing key σ_w, the 128-bit identity ID_w, and the 256-bit public keys X_w and Y_w.

B. Computation Cost

In the authentication process, the IoT device generates the random nonces r_1, r_2, r_3, and r_4 and does *six* scalar point multiplications to compute the randomized points A, P_1, P_2, and P_3, and the commitments T_1 and T_2. For computing σ_t, the IoT device does *one* hash operation to compute the I_a. For verifying the authentication token of the IoT gateway, the IoT device does *two* hash operations, *three* scalar point multiplications, and *two* point additions. Additionally, the IoT device does one hash operation and *one* scalar multiplication for computing the session key.

On the other side, the IoT gateway generates the random nonce r_5 and performs *one* scalar point multiplication to compute W. Moreover, the IoT gateway does *eight* scalar point multiplications and *four* point additions to verify the authentication request of the IoT device. For computing the session key, the IoT gateway does *one* scalar multiplication. During the authentication process, the IoT gateway does *three* hash operations to compute the I_a, I_g, and session key.

C. Communication Overhead

The IoT device initiates the authentication process by sending "Hello" message along with a 2×128-bit randomized EC point A. The IoT gateway responds with a 9×128-bit message $<W, ID_w, X_w, Y_w, \sigma_z>$. In turn, the IoT device replys with 16×128-bit message $<P_1, P_2, P_3, \sigma_t, T_1, T_2, s_1, s_2>$. Thus, in total, the communication overhead between the IoT device and the IoT gateway is 27 $\times$ 128 bits (i.e., 432 bytes).

Compared with other protocols in [26], [28], [30], [31], [32], [33], and [34], MAPFS has the highest communication overhead. However, protocols in [26], [28], and [34] which require 96, 64, 128 bytes, respectively, do not offer MA. Meanwhile, protocols in [30], [31], [32], and [33] require 192, 192, 352, and 240 bytes, respectively, but fail to ensure the IoT unlinkability from the serving IoT gateway, as seen in Table I. This creates a vulnerability that allows an external attacker to compromise the IoT gateway and profile the IoT device.

D. Execution Time

In our comparison, we consider the average time required by each operation as shown in Table IV and the number of the required cryptographic operatios on both the IoT device and gateway sides as reported in Table V. Note that, we neglect modular operations (i.e., multiplication and addition) as they require microsecond execution time. For measuring the average execution time, we use a Raspberry Pi 4 Model B/8GB embedded with a 1.5 GHz 64-bit Quad-core ARM Cortex-A72 processor running the Raspbian 64-bit operating system. We run the cryptographic primitives for 1000 times to compute the average execution time. Moreover, we illustrate the variations of our measurements in a box plot in Fig. 5. For more resource-constrained IoT devices that are beyond the Raspberry Pi capabilities, the Arm Cortex M0 48 MHz ATECC508A HW accelerated as in [35] and [52] can be considered. It offers 0.113 ms AES timing, 0.361 ms Hash timing,

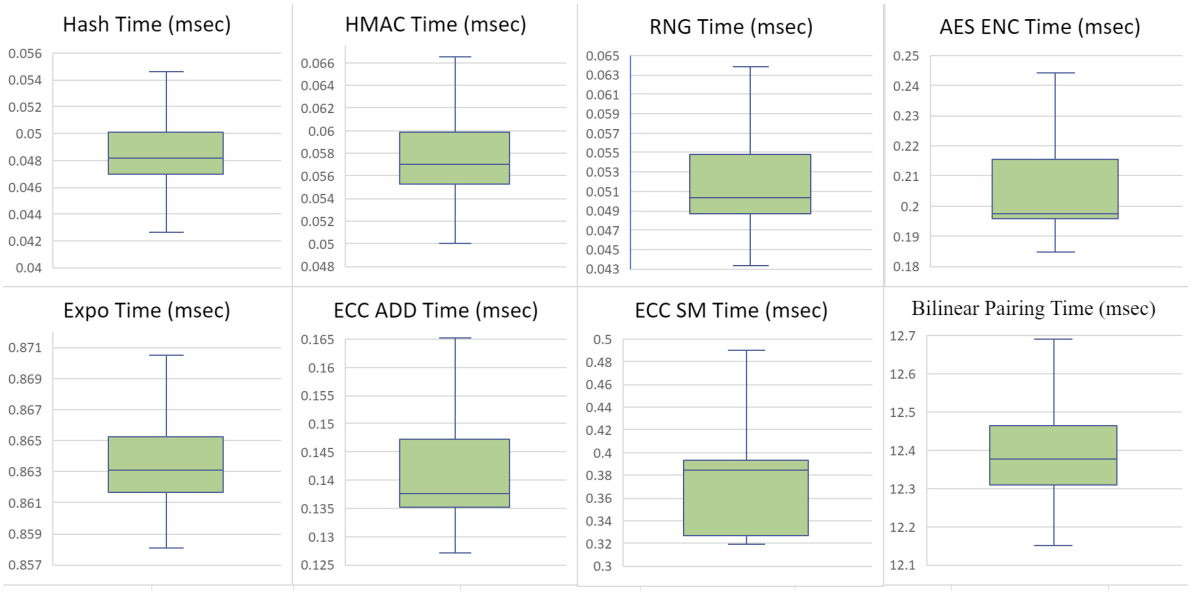

Fig. 5. Box plot for the overhead timing of the cryptographic primitives.

TABLE V
PERFORMANCE COMPARISON BASED ON THE COMPUTATION COMPLEXITY

Protocol	Operations on IoT and Gateway Sides	Total Time (msec)
[28]	4 T_e + T_b	15.93
[32]	9 T_{sm} + 4 T_{pa} + 1 T_e+ 1 T_b + 10 T_h	17.737
[33]	5 T_{sm} + 3 T_{pa} + 1 T_e+ 1 T_b + 11 T_h	16.1627
[30]	7 T_{sm} + 1 T_{pa} + 1 T_e+ 1 T_b + 12 T_h + 2 T_s	17.1
[31]	T_e + 6 T_{sm} + 3 T_b	40.44
[34]	4 T_{sm} + T_e + 1 T_b	14.8
MAPFS	5 T_{rng} + 6 T_{pa} + 20 T_{sm} + 7 T_h	8.8899

0.722 ms HMAC timing, and 2 ms random number generator timing.

We show the total computation-overhead time of the used cryptographic primitives in our protocol MAPFS compared to other related protocols, [28], [30], [31], [32], [33], [34] in Fig. 6. MAPFS has the lowest computation time compared to other protocols and it provides the unlinkability of the IoT request with respect to the IoT gateway beside other security properties.

Our prototype is available as open-source on the GitHub repository https://github.com/LabCryptoLab/MAPFS. This repository includes the Python implementation of the different cryptographic primitives using the bplib, fastecdsa, and crypto libraries for the bilinear pairing, EC operations, and encryption systems, respectively. Additionally, the repository includes a socket programming implementation to simulate the flow of messages between the IoT device and the IoT gateway. The client, a Raspberry Pi 4, played the role of the IoT device, and the server, an Intel laptop 11th Gen Core i7-11800H clocked at 2.3 GHz with 16 GB RAM, acted as the IoT gateway.

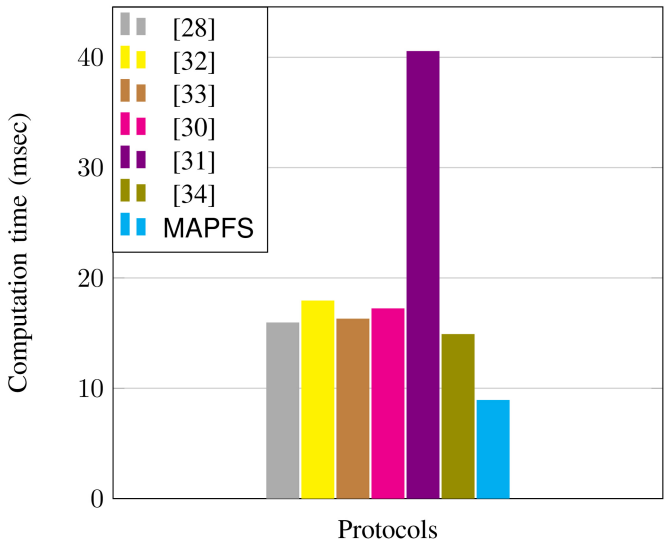

Fig. 6. Evaluations of the overhead associated with cryptographic primitives in schemes [28], [30], [31], [32], [33], [34] and MAPFS.

VIII. CONCLUSION AND FUTURE WORK

In this article, we proposed MAPFS, a privacy-preserving MA protocol between the IoT device and the IoT gateway for computation offloading services in the IoT–Edge–Cloud paradigm. MAPFS achieves anonymity of the IoT device, session unlinkability, and perfect forward secrecy for the established session key with revocation ability for the misbehaving IoT device. Our protocol distinguishes itself from certificate-based anonymous authenticated schemes in that anonymity

can be achieved without additional communication or storage overheads.

MAPFS makes use of ECC for achieving an efficient 128-bit security level. To achieve anonymity of the IoT device and unlinkability property, MAPFS randomizes the authentication token of the IoT. Moreover, it makes use of ZKP to prove the knowledge of the random nonce that binds the authentication token to MAPFS published public parameters. We have formally proved that under intractable ECDLP and ECDDH, MAPFS is MA secure and ensures the secrecy of the session key. Moreover, we have analyzed the protocol's unlinkability, perfect forward secrecy, and backward secrecy. Furthermore, we evaluated MAPFS in terms of storage requirement, communication overhead, and computation cost requirements. Finally, we compared the execution time of our protocol with other closely related protocols.

Finally, it should be noted that in MAPFS, IoT devices register with the KGC to obtain their signing keys. For future work, we plan to investigate registration techniques that better fit the distributed nature of the edge computing paradigm during the registration phase.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[2] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, 2016, pp. 20–26.

[3] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and privacy for cloud-based IoT: Challenges," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 26–33, Jan. 2017.

[4] S. Uludag, K.-S. Lui, W. Ren, and K. Nahrstedt, "Secure and scalable data collection with time minimization in the smart grid," *IEEE Trans. smart grid*, vol. 7, no. 1, pp. 43–54, Jan. 2016.

[5] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle, "Towards viable certificate-based authentication for the Internet of Things," in *Proc. 2nd ACM Workshop Hot Topics Wireless Netw. Security Privacy*, 2013, pp. 37–42.

[6] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Proc. Adv. Cryptol.: 7th Int. Conf. Theory Appl. Cryptol. Inf. Security Gold Coast*, 2001, pp. 552–565.

[7] D. Chaum and E. Van Heyst, "Group signatures," in *Proc. Adv. Cryptol. EUROCRYPT'91: Workshop Theory Appl. Cryptogr. Technol.*, 1991, pp. 257–265.

[8] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. Asiacrypt*, vol. 2894, 2003, pp. 452–473.

[9] S. Li, T. Zhang, B. Yu, and K. He, "A provably secure and practical PUF-based end-to-end mutual authentication and key exchange protocol for IoT," *IEEE Sensors J.*, vol. 21, no. 4, pp. 5487–5501, Feb. 2021.

[10] B. Ying and A. Nayak, "Lightweight remote user authentication protocol for multi-server 5G networks using self-certified public key cryptography," *J. Netw. Comput. Appl.*, vol. 131, pp. 66–74, Apr. 2019.

[11] Z. Xu, C. Xu, W. Liang, J. Xu, and H. Chen, "A lightweight mutual authentication and key agreement scheme for medical Internet of Things," *IEEE Access*, vol. 7, pp. 53922–53931, 2019.

[12] K.-H. Wang, C.-M. Chen, W. Fang, and T.-Y. Wu, "On the security of a new ultra-lightweight authentication protocol in IoT environment for RFID tags," *J. Supercomput.*, vol. 74, no. 1, pp. 65–70, 2018.

[13] H. Chung, K.-C. Choi, and M.-S. Jun, "A design of key agreement scheme between lightweight devices in IoT environment," in *Proc. Adv. Comput. Sci. Ubiquitous Comput.*, 2016, pp. 224–229.

[14] M. Turkanović, B. Brumen, and M. Hölbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion," *Ad Hoc Netw.*, vol. 20, pp. 96–112, Sep. 2014.

[15] M. Seifelnasr, M. Nakkar, A. Youssef, and R. AlTawy, "A lightweight authentication and inter-cloud payment protocol for edge computing," in *Proc. IEEE 9th Int. Conf. Cloud Netw. (CloudNet)*, 2020, pp. 1–4.

[16] M. Seifelnasr, R. AlTawy, and A. Youssef, "Efficient inter-cloud authentication and micropayment protocol for IoT edge computing," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4420–4433, Dec. 2021.

[17] J. Y. Chun, J. Y. Hwang, and D. H. Lee, "A note on leakage-resilient authenticated key exchange," *IEEE Trans. Wireless Commun.*, vol. 8, no. 5, pp. 2274–2279, May 2009.

[18] M. M. Fouda, Z. M. Fadlullah, N. Kato, R. Lu, and X. S. Shen, "A lightweight message authentication scheme for smart grid communications," *IEEE Trans. Smart grid*, vol. 2, no. 4, pp. 675–685, Dec. 2011.

[19] D. Hankerson, A. Menezes, and S. Vanstone, "Elliptic curve arithmetic," in *Guide to Elliptic Curve Cryptography*. New York, NY, USA: Springer, 2004.

[20] C.-T. Li, C.-Y. Weng, and C.-C. Lee, "An advanced temporal credential-based security scheme with mutual authentication and key agreement for wireless sensor networks," *Sensors*, vol. 13, no. 8, pp. 9589–9603, 2013.

[21] W. Shi and P. Gong, "A new user authentication protocol for wireless sensor networks using elliptic curves cryptography," *Int. J. Distrib. Sensor Netw.*, vol. 9, no. 4, 2013, Art. no. 730831.

[22] Y. Choi, D. Lee, J. Kim, J. Jung, J. Nam, and D. Won, "Security enhanced user authentication protocol for wireless sensor networks using elliptic curves cryptography," *Sensors*, vol. 14, no. 6, pp. 10081–10106, 2014.

[23] C.-C. Chang and H.-D. Le, "A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 1, pp. 357–366, Jan. 2016.

[24] X. Li, J. Peng, J. Niu, F. Wu, J. Liao, and K.-K. R. Choo, "A robust and energy efficient authentication protocol for industrial Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1606–1615, Jun. 2018.

[25] P. Tedeschi, S. Sciancalepore, A. Eliyan, and R. Di Pietro, "LiKe: Lightweight certificateless key agreement for secure IoT communications," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 621–638, Jan. 2020.

[26] N. Gayathri, G. Thumbur, P. R. Kumar, M. Z. U. Rahman, P. V. Reddy, and A. Lay-Ekuakille, "Efficient and secure pairing-free certificateless aggregate signature scheme for healthcare wireless medical sensor networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 9064–9075, Oct. 2019.

[27] J. Shen, Z. Gui, S. Ji, J. Shen, H. Tan, and Y. Tang, "Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks," *J. Netw. Comput. Appl.*, vol. 106, pp. 117–123, Mar. 2018.

[28] A. Karati, S. H. Islam, and M. Karuppiah, "Provably secure and lightweight certificateless signature scheme for IIoT environments," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3701–3711, Aug. 2018.

[29] D. Abbasinezhad-Mood and M. Nikooghadam, "An anonymous ECC-based self-certified key distribution scheme for the smart grid," *IEEE Trans. Ind. Electron.*, vol. 65, no. 10, pp. 7996–8004, Oct. 2018.

[30] Y. Li, Q. Cheng, X. Liu, and X. Li, "A secure anonymous identity-based scheme in new authentication architecture for mobile edge computing," *IEEE Syst. J.*, vol. 15, no. 1, pp. 935–946, Mar. 2021.

[31] Y. Jiang, K. Zhang, Y. Qian, and L. Zhou, "Anonymous and efficient authentication scheme for privacy-preserving distributed learning," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2227–2240, 2022.

[32] X. Jia, D. He, N. Kumar, and K.-K. R. Choo, "A provably secure and efficient identity-based anonymous authentication scheme for mobile edge computing," *IEEE Syst. J.*, vol. 14, no. 1, pp. 560–571, Mar. 2020.

[33] X. Jia, M. Luo, K.-K. R. Choo, L. Li, and D. He, "A redesigned identity-based anonymous authentication scheme for mobile edge computing," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 10108–10120, Jun. 2022.

[34] J. Chen, L. Wang, M. Wen, K. Zhang, and K. Chen, "Efficient certificateless online/offline signcryption scheme for edge IoT devices," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8967–8979, Jun. 2022.

[35] M. Nakkar, R. AlTawy, and A. Youssef, "Lightweight broadcast authentication protocol for edge-based applications," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11766–11777, Dec. 2020.

[36] J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*, vol. 1. New York, NY, USA: Springer, 2008.

[37] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.

[38] M. Bellare, "A note on negligible functions," *J. Cryptol.*, vol. 15, no. 4, pp. 271–284, 2002.

[39] C.-P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991.

[40] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Conf. Theory Appl. Cryptogr. Technol.*, 1986, pp. 186–194.
[41] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in *Proc. Conf. Theory Appl. Cryptol.*, 1989, pp. 239–252.
[42] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran, "The role of edge computing in Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 11, pp. 110–115, Nov. 2018.
[43] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Proc. Int. Conf. Theory Appl. Cryptogr. Technol.*, 2001, pp. 453–474.
[44] L. Wei, J. Cui, H. Zhong, I. Bolodurina, and L. Liu, "A lightweight and conditional privacy-preserving authenticated key agreement scheme with multi-TA model for fog-based VANETs," *IEEE Trans. Depend. Secure Comput.*, vol. 20, no. 1, pp. 422–436, Jan./Feb. 2023.
[45] S. H. Islam, "Provably secure dynamic identity-based three-factor password authentication scheme using extended chaotic maps," *Nonlin. Dyn.*, vol. 78, no. 3, pp. 2261–2276, 2014.
[46] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *J. Cryptol.*, vol. 13, no. 3, pp. 361–396, 2000.
[47] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Proc. Int. Conf. Theory Appl. Cryptogr. Technol.*, 1996, pp. 387–398.
[48] C. Cremers and M. Feltz, "Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal," in *Proc. Eur. Symp. Res. Comput. Security*, 2012, pp. 734–751.
[49] C.-C. Lee, M.-S. Hwang, and I.-E. Liao, "Security enhancement on a new authentication scheme with anonymity for wireless environments," *IEEE Trans. Ind. Electron.*, vol. 53, no. 5, pp. 1683–1687, Oct. 2006.
[50] "Standards for efficient cryptography." Accessed: Jan. 20, 2022. [Online]. Available: https://www.secg.org/sec2-v2.pdf/
[51] "bplib 0.0.6." Accessed: Nov. 26, 2021. [Online]. Available: https://pypi.org/project/bplib/
[52] "Wolfssl.com." Accessed: Nov. 26, 2021. [Online]. Available: https://www.wolfssl.com/docs/benchmarks/

Mohamed Seifelnasr (Student Member, IEEE) received the B.Sc. degree in communication, electronics and computer engineering from Helwan University, Cairo, Egypt, in 2012, and the M.Sc. degree in computer science from Huazhong University of Science and Technology, Wuhan, Hubei, China, in 2018. He is currently pursuing the Ph.D. degree with the Concordia Institute for Information Systems Engineering (CIISE), Concordia University, Montreal, QC, Canada.

His current research interests include Internet-of-Things and edge computing security.

Riham AlTawy (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees from AAST, Alexandria, Egypt, in 2005 and 2008, respectively, and the Ph.D. degree from Concordia University, Montreal, QC, Canada, in 2016.

She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada. Previously, she was an NSERC Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. Her research interests focus on IoT security, blockchains, and lightweight cryptography.

Amr Youssef (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees from Cairo University, Cairo, Egypt, in 1990 and 1993, respectively, and the Ph.D. degree from Queens University, Kingston, ON, Canada, in 1997.

He is currently a Professor with the Concordia Institute for Information Systems Engineering (CIISE), Concordia University, Montreal, QC, Canada. Before joining CIISE, he worked with Nortel Networks, Ottawa, ON, Canada; the Center for Applied Cryptographic Research, University of Waterloo, Waterloo, ON, Canada; IBM, New York, NY, USA; and Cairo University. He has more than 230 referred journal and conference publications in areas related to his research interests. He also served on more than 60 technical program committees of cryptography and data security conferences. His research interests include cryptology, cybersecurity, and cyber–physical systems security.

Prof. Youssef was the Co/Chair of Africacrypt 2013 and Africacrypt 2020, the conference Selected Areas in Cryptography (SAC 2014, SAC 2006, and SAC 2001).

Essam Ghadafi (Senior Member, IEEE) received the B.Sc. degree in computer science from the University of Tripoli, Tripoli, Libya, in 1998, and the M.Sc. degree in advanced computing and the Ph.D. degree in cryptography and information security from the University of Bristol, Bristol, U.K., in 2008 and 2012, respectively.

He worked as a Postdoctoral Researcher with the University of Bristol and University College London, London, U.K. He is currently a Senior Lecturer of Cyber Security with Newcastle University, Newcastle upon Tyne, U.K. He led and participated in various research projects. His experience also includes supervising the Ph.D. students. His research interests include cryptography and information security.