GASE: A Lightweight Group Authentication Scheme With Key Agreement for Edge Computing Applications

Mouna Nakkar, *Member, IEEE*, Riham AlTawy, *Senior Member, IEEE*, and Amr Youssef, *Senior Member, IEEE*

***Abstract*—Motivated by the fact that mass authentication is one of the desirable security features in the edge computing paradigm, we propose a lightweight group authentication protocol with a session key-agreement. Most of the previously proposed group authentication schemes (GASs) are heavyweight and do not support multiple authentications or key-agreement. On the other hand, our protocol, which is based on secret sharing scheme and aggregated message authentication code, is lightweight and provides multiple asynchronous authentications. Furthermore, we implement a simple key refreshing mechanism in which, in each session, a new session-key between an Internet of Things node and the authenticating server is established without the need for redistributing new shares. Our security analysis includes proving that our protocol provides group authentication, message forward secrecy, and prevents several attacks. Additionally, we present a formal automated verification using Verifpal tool. Furthermore, we show that our scheme has better performance than other relative schemes in terms of communication complexity, secret-share redistribution, and session key derivations.**

***Index Terms*—Aggregated message authentication code (Agg-MAC), edge computing (EC), group authentication, massive machine-type communication (mMTC), secret sharing.**

I. Introduction

THE PROLIFERATION rate of the Internet of Things (IoT) is exponentially on the rise. The projected number of devices connected to the Internet is expected to triple to reach 25 Billion in 2030, [1]. The IoT usage ranges from personal smart appliances to prime industry verticals such as smart grids, water supply and waste management, transportation, gas, retail and wholesale, and commercial businesses. Indeed, with this migration to the smart life, many concerns are raised. Most important ones are related to performance and security. For instance, if million-nodes requesting authentication at the same time from the same server, this would potentially create congestion, communication, and security problems.

Manuscript received 30 August 2021; revised 20 February 2022 and 16 April 2022; accepted 30 August 2022. Date of publication 6 September 2022; date of current version 22 December 2022. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). *(Corresponding author: Riham AlTawy.)*

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

Digital Object Identifier 10.1109/JIOT.2022.3204335

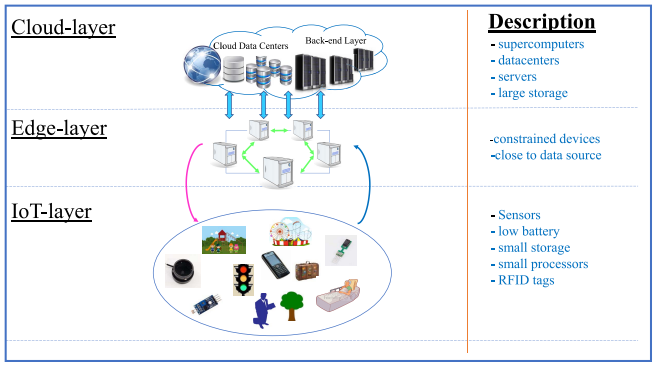

Fig. 1. Three-tier cloud-edge-IoT framework.

Edge computing (EC) is a recently developing network paradigm that comes about primarily to address performance issues in the IoT era. The main idea is to push data processing toward the source of the data rather than the traditional centralized approach. This decentralization mechanism is proved to improve performance and reduces cloud congestion. Security, on the other hand, still remains a major concern for this new paradigm. This is primarily because the traditional security solutions are mostly heavyweight and not suitable for the low-end IoT devices. Additionally, new security goals and threats are constantly arising with this new developing paradigm. For example, deploying security solutions in the decentralized edge framework which involves several edge servers increases the possibility of edge-servers compromise when compared to the deployment of the same security solutions in one centralized cloud computing server [2].

The architecture for the EC paradigm as presented in recent surveys [3], [4], [5] is composed of three layers, the cloud layer, the edge layer, and the low-end devices IoT layer as shown in Fig. 1. The cloud layer is the far-end layer which contains all servers and distanced data centers, and has large storage capacity. The edge layer, on the other hand, is the layer that is close to the data source. In the literature, the terms fog computing and EC are sometimes used interchangeably. However, we use the definition provided by [5] and consider the edge to be the entities that are close to the source of the data. The edge layer contains devices and gateways that have moderate computational powers, storage capacity, and

battery life time. On the other hand, the IoT layer is the layer that contains all low-end nodes, such as sensors and radio frequency identification (RFID)s tags and readers.

Group authentication is emerging to be an exceptionally promising security solution for the three-tier cloud-edge-IoT computing paradigm in which massive machine-type communication (mMTC) is an expected scenario. Specifically, for the above-mentioned example of million-nodes connected to one server, it is extremely efficient to authenticate all these nodes at the same time instead of one at a time. Because of this, new proposals focusing on group authentication are recently presented in the literature. However, the existing group authentication schemes (GASs) are heavyweight with large communication overhead. Furthermore, most of these solutions are not applicable for multiple authentication and do not support any mechanism for refreshing the node shares. We note here that the term multiple authentication in GAS refers to a scheme that supports multiple group authentication without redistributing new shares or rerunning the set-up phase as described in [6], [7], [8], and [9].

A. Motivation

EC paradigm is one of the cloud-extended paradigms developed recently to solve the current cloud computing centralization issues. In addition to decentralization, EC offers abundance of advantages, such as meeting delay-time requirements, efficiency, computation offloading, storage offloading, scalability, mobility, and others [5]. These advantages prompted many designers to adopt the EC paradigm in smart applications. However, security remains a major concern for EC applications. This is because achieving security goals such as authenticity in smart edge applications could be a challenging task. For example, consider a common situation in EC smart applications of having a large number of IoT-nodes connected to a single edge server, say thousands nodes connected to one EC server. In this scenario, a decentralized mass authentication is more efficient than authenticating one node at a time with the main authentication server (AS). Indeed, this mass authentication reduces communication bandwidth on both AS and IoT-nodes. Also, the computational burden on the AS is reduced significantly with mass authentication. Furthermore, in group authentication, each group member is able to authenticate other group members independently of the main edge server and without any certificates. Thus, this also reduces computational burden on the low-end IoT-nodes. One of the most compelling examples for mass authentication is in the ad hoc network in which the communication with the base station is very expensive to the low-end limited-battery devices [10]. Thus, mass authenticating the nodes at the edge is more efficient. Group and mass authentication are currently gaining large attention in the research community [6], [7], [8], [9], [11], [12].

B. Contribution

Our contribution can be summarized as follows.

1) We propose a lightweight GAS based on secret-sharing, and symmetric-key cryptography. Our GAS at the edge (GASE) provides asynchronous multiple authentication, key agreement, and forward secrecy. Furthermore, we provide an efficient session key refreshing mechanism without redistribution of the shares.
2) We formally prove that our protocol provides group and message authenticity. Also, we verify our protocol security using Verifpal automated verification tool [13].
3) We implemented related security primitives/operations on Raspberry Pi 4 Model B/8GB/Broadcom-BCM2711, Quad-core Cortex-A72-1.5GHz (ARM v8) 64-bit SoC processor, and applied the experimental execution time for each operation to compare between our protocol and others. The results show that our protocol is computationally competitive when compared to related protocols.

The remainder of this article is organized as follows. In Section II, we present the related work, and in Section III, we briefly review the required background for our proposal. Section IV presents our system model, threat model, and design objectives. We present our proposed protocol in Section V. In Sections VI and VII, we present our security analysis and computational requirements, respectively. Finally, our conclusion is provided in Section VIII.

II. Related Work

In this section, we provide a brief summary of the development of GASs in the literature, and in a later section, we compare the most related research to our proposal.

A. Group Authentication by Secret-Sharing

The foundation of group secret sharing schemes is the (n, t) Shamir secret sharing (SSS) scheme, [14], [15], where t is the threshold and n is the number of group members. In this scheme, group members must combine their secret shares to recover the group secret [16]. Based on SSS, Harn proposed the first GAS in 2013, [6]. GAS allows members to authenticate each other in a many-to-many group authentication style. In this paper, Harn presented three (n, t, m) GASs based on SSS: 1) basic; 2) asynchronous; and 3) asynchronous multiple authentication. Harn's schemes 1 and 2 are basic SSS and allow for only one-time group authentication. However, Harn's scheme-3 allows a group of n members to authenticate each other multiple times, where m is the number of participants in the authentication process and t is the number of secure users. Harn's scheme-3 prevents t-users collusion and allows many-to-many authentications. Although flexible and based on SSS, Harn's scheme-3 is heavyweight in terms of computations. Precisely, each member must compute an exponential modular operation which is equivalent to RSA operation in addition to the Lagrange's formula computed in $GF(q)$ field.

To reduce computational complexity, [7], [8], [9] propose their GASs based on Harn's scheme-1 or 2. For example, Aydin's *et al.* [7] used elliptic curve cryptography (ECC) to securely reveal the node's share multiple times. Thus, each user in Aydin's scheme computes only one ECC multiplication in a centralized group authentication setting. On the other hand, Li *et al.* [8] utilized Harn's scheme-2 and ECC-pairing

for group authentication and key agreement, respectively, for the LTE network. Similarly, Chien [9] used Harn's scheme-2, ECC, and pairing for multiple-group authentications. However, these proposed GAS schemes, [7], [8], [9], do not have any mechanism to refresh the secret session keys. Indeed, the GAS schemes proposed by [7], [8], [9] are closely related to our scheme, and these schemes provide key agreement; however, in these schemes the session key is not refreshed, and thus, the forward secrecy property is not ensured. In our study, we propose a lightweight GAS with multiple authentication and a key refreshing mechanism that provides forward secrecy.

Similar to the above-proposed schemes, other researchers propose group authentication based on threshold cryptography or Lagrange's interpolation. For example, Shabisha *et al.* [17] proposed a fog-centered group authentication in which they use ECC, Schnorr signature scheme, and Lagrange's polynomial for group authentication and group key construction, respectively. On the other hand, Kaya and Selçuk [18], scheme relied on Paillier threshold cryptography and requires public key cryptography (PKC) encryption and carries large computations. Similarly, Yang *et al.* [19] used bilinear pairing and threshold cryptography to implement a delegated authentication in the vehicle network framework; namely, the scheme allows the edge servers to collaboratively authenticate vehicles. Because of the computational complexity, Yang's scheme is not suitable for low-end IoT-devices.

B. Group Authentication by Aggregated-MAC

Aggregated message authentication code (Agg-MAC) [10], is excessively used for group authentication in the mMTC paradigm, such as long-term evolutionary-advanced (LTE-A) or fifth-generation (5G) network. In these network paradigms, group authentication is an essential security primitive. For example, Lai *et al.* [11] proposed an LGTH group authentication designed for machine-type communication (MTC) in the LTE networks. Specifically, the mobile management entity (MME) aggregates all MAC codes from all machine-type communication devices (MTCDs) and sends them to the home subscriber server (HSS) for authentication. Similarly, Lai *et al.* [12] and Qiu and Ma [20] used the Agg-MAC for third-generation partnership project (3GPP) and 6LoWPAN networks, respectively.

C. PUF-Based Group Authentication

Researchers utilized physical unclonable function (PUF) devices for group authentication, group key agreement, and communication. For example, Yildiz *et al.* [21] proposed a PLGAKD scheme which is based on PUF, Chinese reminder theorem (CRT), and factorial tree security primitives to ensure group authentication and key distribution. On the other hand, Ren *et al.* [22] proposed a GAS for the narrowband IoT 5G framework which is simply based on PUF, hash, and XOR functions. Other PUF-based proposals [23], [24] are concerned with group communication and key distribution; however, they do not provide group authentication. A recent proposal by Chen *et al.* [25] utilized PUF for mutual authentication leveraging SSS for availability and reliability. However, the scheme is not for group authentication or group communication.

D. Group Authentication Based on Multivariate Polynomial

Other types of GASs proposed in the literature are based on symmetric multivariate-polynomial security primitive, [26], [27], [28]. Unlike the SSS and Harn's schemes, the multivariate polynomial is in the form of $f(x_1, x_2, \ldots, x_n)$ such that if any of the variables, say x_i and x_j, are interchanged, the polynomial remains the same. For example, Li *et al.* [26] proposed a secret sharing scheme based on bivariate polynomial in which the scheme reduces the group manager computational complexity. However, the communication complexity remains the same as Harn's. Similarly, Cheng *et al.* [28] proposed a multivariate-polynomial scheme for group membership authentication for the wireless sensor network (WSN) which reduces the authentication from $O(n^2)$ to $O(n)$ where n is the group size.

E. Other Approaches Based on ECC

There several proposals for massive MTCD authentication in the literature. For example, Cao *et al.* [29] proposed a massive narrowband fast authentication scheme for the 3GPP 5G network. Their scheme is based on the certificate-less aggregate *Signcryption* PKC scheme. Similarly, Lai *et al.* [30] and Basudan [31] used group authentication and key agreement based on ECC-cryptography and bilinear pairing for the LTE and 5G framework, respectively, while [32] base their group authentication on Chebyshev chaotic maps-based cryptography primitive. Chien [33] proposed an aggregated authentication-key-agreement (AKE) scheme for the group-oriented-range-bound which provides lesser authentication overhead when compared to its counterparts. On the other hand, group signatures schemes, such as [34], [35], and [36] are heavyweight with large overhead. Thus, we base our protocol on secret-sharing and Agg-MAC only.

Similarly, other GASs proposed in the literature are based on CRT [37], [38], [39]. However, because of the computational complexity and large communication overhead, we base our protocol on SSS instead of CRT. Table I shows summary of related protocols.

Finally, in terms of EC authentication protocols, only few proposals are found in [40], [41], and [42]. However, none of the cited proposals addresses edge group authentication.

F. Distance-Bounding Protocols

Another relevant topic to the edge-IoT layer authentication is distance-bounding protocols [43]. A distance-bounding protocol is a cryptographic mechanism that gives the verifying party an upper-bound of the physical distance to the prover. For instance, a verification entity at the entrance of a building would like to ensure that the communicating individual is only few meters away. This occurs by a bit exchange between the prover and verifying party, and the difference between delay times gives the verifying party an estimate of the distance between them. In some RFID applications, it is very important

TABLE I
RELATED WORK SUMMARY

Reference	Application & Utilized Schemes	Security Properties and Features	Security Primitives
This work	– edge computing paradigm – GAS scheme – key agreement	– group authentication & confidentiality, – forward secrecy – key update mechanism – multiple group authentication	– multi-secret sharing – aggregated MAC
Harn [6]	– three GAS schemes – no key agreement	– group authentication – multiple authentication	– SSS scheme – exponential modular operation
Aydin [7]	– edge computing paradigm – GAS scheme – key agreement	– group authentication & confidentiality – no secret update mechanism	– SSS sharing – ECC cryptography
Li [8]	– LTE network – GAS scheme – key agreement	– group authentication & confidentiality, – single authentication	– Harn's scheme 2 – ECC cryptography – MAC
Chien [9]	– GAS scheme – no key agreement	– group authentication, – single authentication	– SSS scheme – bi-linear pairing
Shabisha [17]	– fog computing paradigm – group key agreement – pairwise fog-node key agreement	– group authentication & confidentiality, – single authentication	– ECC Cryptography – Schnorr signature – Lagrange interpolation
Yang [19]	– edge computing paradigm – de-centralized vehicle network	– delegated mutual authentication – node joining/revoking mechanism – fast handover mechanism	– bi-linear pairing – threshold cryptography – Identity-based signature
LGTH [11]	– LTE network – massive machine type communication	– massive MTC authentication – single authentication	– aggregated MAC
GLARM [12]	– 3GPP network – massive machine-2-machine communication	– massive M2M authentication – single authentication	– aggregated MAC
Qiu [20]	– 6LoWPAN network – Proxy Mobile IPv6 – group mobility	– group authentication – group handover – single authentication	– aggregated MAC – encryption – hash + xor
PLGAKD [21]	– group communication for IoT framework – smart lighting application – key distribution	– group authentication – node joining/revoking mechanism	– PUF-based devices – CRT + encryption – factorial tree + hash
Ren [22]	– 5G network – NB-IoT framework – key agreement	– massive authentication – secure data transmission	– PUF-based devices – truncated & aggregated authentication code – Hash + XORs
Li [26]	– GAS scheme – no key agreement	– group authentication & communication – single authentication	– SSS scheme – Bi-variate polynomial
Cheng [28]	– many-to-many group communication – WSN paradigm – group key agreement	– membership authentication – group forward/backward secrecy – updated group key mechanism	– SSS scheme – multi-variate polynomial – hash
Cao [29]	– 5G network – massive narrow-band IoT (NB-IoT)	– access authentication & data transmission – secure data transfer	– certificateless aggregate signcryption, – encryption
SE-AKA [30]	– LTE network – group temporary key (GTK) agreement	– group authentication – single authentication	– ECDH exchange – MAC
LEGA [31]	– 5G network – mMTC	– group authentication & confidentiality – forward/backward secrecy – single authentication	– bi-linear pairing – aggregate certificateless signature mechanism
LSSA [32]	– 3GPP mobile devices – mMTC – key agreement	– access authentication – forward/backward secrecy – single authentication	– Chebyshev chaotic maps – encryption – MAC
Chien [33]	– group-oriented-range-bound – Authenticated-key-agreement	– aggregated & delegated authentication – homogeneous trust & authorization	– bilinear pairing – hash function

to identify the location of the tag; otherwise, the system may be subject to mafia or relay attacks, [44], [45], [46], [47].

III. PRELIMINARIES

In what follows, we briefly present the security primitives used in our proposed scheme.

A. Secret Sharing Schemes

In the (n, t) SSS scheme [14], where t represents the threshold and n is the number of group members, a trusted dealer (D) generates a $(t-1)$-degree polynomial $d(x) = k_0 + k_1 x^1 + \cdots + k_{t-1} x^{t-1}$, such that $d(0) = k_0$ is the secret. During setup, the dealer distributes $(x_i, y_i = d(x_i))$ tokens to each group member i through a secure channel. To recover the secret, at least t shareholders reveal their tokens $d(x_i)$ to compute the Lagrange's interpolation formula. The formal definition of a secret sharing scheme is as follows.

Definition 1: A secret sharing scheme over $\mathbb{Z}_q$ is composed of two algorithms, namely, $\mathcal{G}$ and $\mathcal{C}$.

1) $\mathcal{G}$ is a probabilistic algorithm that generates t-out-of-n shares of k. $\mathcal{G}$ is invoked as $\mathcal{G}(n, t, k) \xrightarrow{R} (s_1, s_2, \ldots, s_n)$ where n is the number of shares, t is the threshold such that $0 < t \leq n$, k is the secret, and s_i is the share for node i.
2) $\mathcal{C}$ is a deterministic algorithm $k \leftarrow \mathcal{C}(s'_1, s'_2, \ldots, s'_t)$. It is invoked to recover k using the Lagrange's interpolation formula.
3) *Correctness:* For every t set of shares of k, $\mathcal{C}(s'_1, s'_2, \ldots, s'_t) = k$.

On the other hand, in the (n, t, β) multisecret sharing schemes, a group of at least t out of n participants share their secret-shadows to recover β secrets. Most of these

schemes [48], [49], [50], [51], [52], [53] are based on SSS scheme [14] and a two-variable one-way function $f(r, s)$ [48]. The properties of the two-variable one-way function, as shown in [49], are as follows: 1) it is easy to compute the function, $f(r, s)$ given any values of r and s; 2) having the knowledge of s and $f(r, s)$, it is hard to find the corresponding r; 3) having the knowledge of r and $f(r, s)$, it is hard to find the corresponding s; 4) without the knowledge of s, it is hard to find $f(r, s)$ for any given value of r; 5) knowing s, finding $f(r_1, s) = f(r_2, s)$ such that $r_1 \neq r_2$ is hard; and 6) given any number of pairs of $f(r_i, s)$ and r_i, finding $f(r', s)$ such that $r' \neq r_i$ is hard.

In our protocol, we utilize Yang's *et al.* scheme [49] with $k = 1$ and the two-variable one-way function to implement a GAS at the edge of the network.

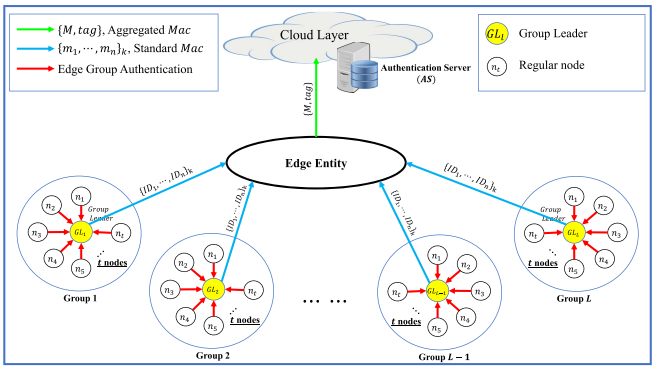

Fig. 2. Group authentication in the three-tier cloud-edge-IoT framework.

B. Aggregated Message Authentication Codes

Katz and Lindell [10] proposed an Agg-MAC, and prove that its security properties are analogous to the security properties of the standard MAC.

Definition 2: Agg-MAC is a set of probabilistic-polynomial-time (PPT) algorithms, namely, (MAC, Agg-MAC, Verify), defined as follows.

1) *MAC:* Let $\kappa \in \{0,1\}^{\lambda}$ be a key with a length equals to the security parameter, λ, and let $\text{msg} \in \{0,1\}^*$ be any arbitrary length message, MAC is constructed from the standard keyed pseudorandom function, $\text{tag} \leftarrow \text{MAC}_\kappa(\text{msg}) = \mathcal{F}_\kappa(\text{msg})$ where $\mathcal{F}$ is a pseudorandom function.
2) *Agg-MAC:* Let (msg_i, d_i) and tag_i be a message/identifier pair and its corresponding tag for node i, respectively. The new message (M, tag) is the aggregation of all l messages, i.e., $M = \{(\text{msg}_1, \text{id}_1), (\text{msg}_2, \text{id}_2), (\text{msg}_3, \text{id}_3), \ldots, (\text{msg}_l, \text{id}_l)\}$, and the corresponding tag is constructed by simply XOR-ing all the messages tags, $\text{tag} = \text{tag}_1 \oplus \text{tag}_2 \oplus \text{tag}_3 \oplus \cdots \oplus \text{tag}_l$.
3) *Verify:* Given the set of all identifiers keys $\{\kappa_1, \kappa_2, \kappa_3, \ldots, \kappa_l\}$, and the message tag pair, (M, tag), the verify algorithm, $\text{Verify}_{(\kappa_1,\text{id}_1),(\kappa_2,\text{id}_2),\ldots,(\kappa_l,\text{id}_l)}(M, \text{tag})$ outputs 1 upon successful authentication; and 0 otherwise.

IV. System Model and Design Objectives

A. System Model

The system model for our proposal is shown in Fig. 2. We have three network layers: 1) the cloud layer; 2) the edge or gateway layer; and 3) the low-end IoT layer. We consider that the group leader (GL) and members communicate in a wireless medium in which each node, including the GL, broadcasts its message to all group members. We further assume that the group members are within the same vicinity and have strong short-range communication protocol such as ZigBee, in which all broadcast messages are accessible to all nodes. The GL authenticates group members and relays their messages to the edge entity using say mobile broadband communication. The details of each entity in our model are described as follows.

1) *Authenticating Server AS:* The authenticating server (AS) is located in the cloud layer. The AS is responsible for the IoT-node and edge entities registrations and system initialization. Specifically, during initialization, the AS randomly selects two long-term secret-shadows for each node in the system and distributes them to each node using a secure channel. These secret-shadows are long-term secrets for each node and are independent from each other. On the other hand, for mass authentication and session key agreement, the AS divides the registered system nodes into L groups such that all group nodes share the same geographical proximity or common ownership. For each group, the AS assigns an edge entity and several related IoT-nodes.
2) *Edge Entity:* The edge entity is a middle intermediate layer between the cloud and the IoT-nodes layer. In our model, one edge entity can be associated with several groups. In the authentication process, the edge entity acts as an aggregator of messages received from each group to the authenticating server. The edge entity has better computational capabilities compared to the IoT-nodes.
3) *GL:* The AS initially groups the registered nodes according to their geographical proximity, and for each group, the AS assigns a GL. The GL selection could be based on different parameters, such as available energy/resources, distance form other nodes, communication range, or density [33], [54]. However, in our system, the AS selects a GL that has an adequate processing, power resources, and mobile broadband communication range. This is because the GL has to be active to authenticate all group members and sends the authenticated IDs to the edge server entity, and this requires sufficient device resources and communication range. Other nodes in the group may have short-range communication like ZigBee. Furthermore, the GL needs to be available and active all the time. In case the GL is inactive, the authentication process is not conducted. Alternatively, the AS may assign another backup GL such that if the main GL is inactive, the backup GL conducts the authentication process.
4) *IoT-Nodes:* In each group, there is number of registered IoT-nodes which are low-end devices with small storage capacity. After authentication, each IoT-node shares a

session key with the AS, and consequently, all messages between AS and the IoT-node gets encrypted with this key.

B. Threat Model

We assume that AS is a trusted entity that communicates with the edge and the IoT-nodes during initialization phase through a secure channel. We also assume that the edge is a trusted gateway. However, the IoT-nodes in our model are vulnerable small devices that are physically accessible to the attacker. If the IoT-node gets compromised, the attacker can access all its secrets. We assume that the adversary is not able to compromise more than $\lfloor (t-1)/2 \rfloor$ nodes. Because the communication between the GL and members is through wireless medium, the adversary may attempt several attacks, such as eavesdropping, intercepting packets, redirecting packets, or injecting packets.

For our group setting, we consider two types of adversaries: 1) an outside adversary and 2) an inside adversary. In what follows, we describe the attacks that could be conducted in our framework.

1) *Outside Adversary Impersonation Attack:* The outside adversary is not a registered node in the system and does not have valid shares. The aim of this adversary is to impersonate a valid registered node in the system; perhaps to receive free access of paid-services. Additionally, the outside adversary may try to access confidential communications between IoT-nodes and the server.
2) *Inside Adversary Impersonation Attack:* The inside adversary, on the other hand, is a registered node in the system and has valid shares; thus, the inside adversary passes the authentication process. However, similar to the outsider, the inside adversary tries to impersonate a valid registered node to perhaps obtain paid-services and send the charges to this valid registered node. The inside attacker may also try to access confidential messages between IoT-nodes and server. We also, assume that both types of adversaries have access to all communications between the cloud, edge, and IoT-devices including the public values from the AS.
3) *Asynchronous-Release Attack:* Harn [6] described an asynchronous-release attack in which the adversary takes advantage of the time of the released shares. Specifically, the adversary may wait for all participants to reveal their shares and recover the $(t-1)$-degree polynomial and reveal its good share afterword. This attack could be conducted by both types of adversaries, the outside attacker and the inside attacker.
4) *IoT-Node Collusion:* Another possible attack for the group secret-sharing setting is the node collusion. Namely, in the $(t-1)$-degree polynomial, the attacker requires t shares to recover the entire polynomial and its secret. The inside or outside attacker may collude with other IoT-nodes to obtain the necessary shares to recover the secret polynomial.
5) *Replay Attack:* Either the outsider or the insider adversary may eavesdrop on the communications between the IoT-nodes, edge entities, and the authenticating server. The attacker may take advantage of unsecured communications and conduct a replay attack in which the adversary rerun old sessions.
6) *Token-Forgery Attack:* The inside or outside adversary may try to forge "good" shares to pass the authentication process and access the services provided by the system.
7) *Overtaking-IoT-Node:* The attacker may try to physically access the IoT-node and obtain all its secrets, including the long-term key and the secret shares.

C. Protocol Goals

The goals of our GASE scheme are listed as follows.

1) *Group Authentication and Confidentially:* The main objective of our proposal is to mass authenticate the IoT-nodes without congesting the authenticating server. Indeed, there could be million of nodes associated with a single AS, and if each IoT-node communicates with the sever at the same time, this creates congestion and large traffic on the server. Thus, our goal is to provide lightweight low communication overhead mass authentication protocol to prevent server congestion and keep the confidentiality of the message exchange between IoT-nodes and AS.
2) *Asynchronous Share-Release:* In this type of share-release, the participants release their shares asynchronously, i.e., not at the same time. This creates a possible asynchronous-release attack as described by Harn [6] in which the attacker waits for all participants to release their shares and releases its good share afterwords. Our goal is to protect against this type of asynchronous-release attack.
3) *Key Agreement:* Most of the GAS schemes proposed in the literature do not support key agreement. This goal aims to realize both primitives within the same protocol.
4) *Updated Session Keys and Forward Secrecy*: We aim to provide the efficient key refreshing mechanism so that AS distributes the secret-shadows to each node only once during the initialization phase. Nevertheless, the secret shares get refreshed with every session without the need to rerun the initialization phase, thus, achieving forward secrecy where old encrypted messages are protected if either the current session key or long-term key is leaked [55].
5) *Lightweight and Efficiency*: Low-end devices at the edge inherently have small computational power and limited memory capacity. Thus, our goal is to present an authentication protocol that is lightweight, requires low storage, and supports low bandwidth communication.

The aforementioned features are not all included in other schemes found in the literature. Table II shows a comparison of the most relevant schemes [6], [7], [8], [9], where Harn-3 refers Harn's Scheme #3.

V. Edge Group Authentication Scheme

In this section, we present our lightweight GAS designed for the EC paradigm. Our protocol is composed of four phases, namely, initialization and setup phase, hashed-shares

TABLE II
COMPARISON WITH RELEVANT SCHEMES

Parameters	Harn-3 [6]	Aydin [7]	Li [8]	Chien [9]	Ours
Initial share delivery	Secure-channel	Secure-channel	Secure-channel	Secure-channel	Secure-channel
Security-primitive	SSS	SSS & ECC	SSS & ECC	SSS & paring	MSS & AggMAC
Arbitrary group leader	✓	✓	✗	✓	✓
Asynchnous share-release	✓	✓	✓	✓	✓
Key agreement	✗	✓	✓	✗	✓
Key update	✗	✗	✗	✗	✓

TABLE III
NOTATIONS

Notation	Description
AS	Authentication Server
GL	Group leader
$\mathcal{A}$	A polynomial-time adversary
s_i	Secret-shadow for node i
ss_i	Second secret-shadow for node i
t	Threshold of secret recovery
n	Total number of nodes in the group
m	Total number of participants
N	Total number of nodes in the system
r	Random number
w	Time window
TS	Time stamp
$f(r, s_i)$	Two-variable one-way function for r and s_i
(x_i, y_i)	a secret share, $y_i = d(x_i) = d(f(r, s_i))$ for node i
(xs_i, ys_i)	a second share, $ys_i = d(xs_i) = d(f(r, ss_i))$
k_0	k-secret, $k_0 = d(0)$
h	Hash of secret, $h = H(k_0)$
θ	Number of polynomials in the scheme
$< data >$	Un-encrypted data packet
$\{message\}_{key}$	Authenticated encryption for $message$ with key
Sk_i	Session key between node i and AS
LTK_i	Long-term key between AS and node i
LTK_iH	Long-term key between AS and node i for HMAC
$AggMAC$	Aggregated Message Authentication Code
GLM_l	group message=$\{(msg_1, id_1), \cdots, (msg_n, id_n)\}$
tag_l	Group $tag_l \leftarrow MAC_k\{GLM_l\}$
$H(.)$	One-way hash function

reveal phase, GL authentication and secret recovery phase, and server authentication phase. The used notations are shown in Table III.

A. Overview

Our main objective is to efficiently mass authenticate the low-end IoT-devices in the three-tier cloud-edge-IoT framework without overloading the authenticating server. To this end, we utilize security primitives related to the secret sharing schemes and the Agg-MAC presented in Section III. Specifically, we divide the total number of registered IoT-nodes in the system N into L groups. Each group has at least t number of secret-shadow holders. In each group, there is one edge entity and a GL as shown in Fig. 2. The authentication process occurs in three stages. First, the GL authenticates all group members using Yang's multisecret sharing scheme. Then, the GL sends the edge entity all group members identifications *IDs* authenticated with a standard MAC. Finally, the edge entity aggregates all tags received from each GL, and sends it to the authenticating server for verification.

B. Grouping and Asynchronous Share-Release

In our model, AS is responsible for grouping the IoT-nodes and edge entities during the initialization and registration phase. The grouping could be based on geographical proximity or common ownership. Thus, the groups may have different sizes, and they may be independent from each other. Nevertheless, the size of a group at any given time must be greater or equals $\geq t$, where t is the degree of the group polynomial. Furthermore, the group size could change upon adding new IoT-nodes or revoking existing IoT-nodes from the group. However, during the authentication process, we require that the group nodes remain static. Specifically, when m participants in a group join the authentication process, we require no changes to the group, namely, no adding nor revoking of nodes. Furthermore, even though, the m participants may reveal their shares asynchronously, our protocol protects against the asynchronous-release attack described by Harn [6]; the details of the asynchronous-release attack are shown in Section VI-B.

C. Initialization and Setup Phase

In our protocol, the AS acts as the secret dealer (D), and initially, it sets the environment for the group authentication. The details are as follows, and a summary is given in Fig. 3.

a. AS randomly generates $2N$ secret-shadows, namely, $(s_1, ss_1), (s_2, ss_2), \ldots, (s_N, ss_N)$, and distributes them securely to each registered IoT device in the system such that each node has two secret-shadows, (s_i, ss_i).
b. AS divides the registered nodes in the system into L groups where each group has one edge entity. The grouping mechanism is described in Section V-B.
c. For each group, AS generates a unique $(t-1)$-degree polynomial in the format

$$d(x) = k_0 + k_1 x^1 + \cdots + k_{t-1} x^{t-1} \mod q \quad (1)$$

where q is a large prime and $0 < k_0, k_1, \ldots, k_{t-1} < q$, and $k_0 = d(0)$ is the secret used for authentication and node session key derivation.
d. AS randomly chooses r and computes the two secrets, $x_i = f(r, s_i)$ and $xs_i = f(r, ss_i)$, and their corresponding $y_i = d(x_i) = d(f(r, s_i))$ and $ys_i = d(xs_i) = d(f(r, ss_i))$ for $\{i = 1, \ldots, N\}$.
e. AS computes $h = H(k_0)$ where $H(\cdot)$ is a one-way hash function.
f. For each group, AS selects the GL for the current session and publishes the following values:

$$<r, \mathrm{ID_{GL}}, (y_1, ys_1), (y_2, ys_2), \ldots, (y_n, ys_n), h>.$$

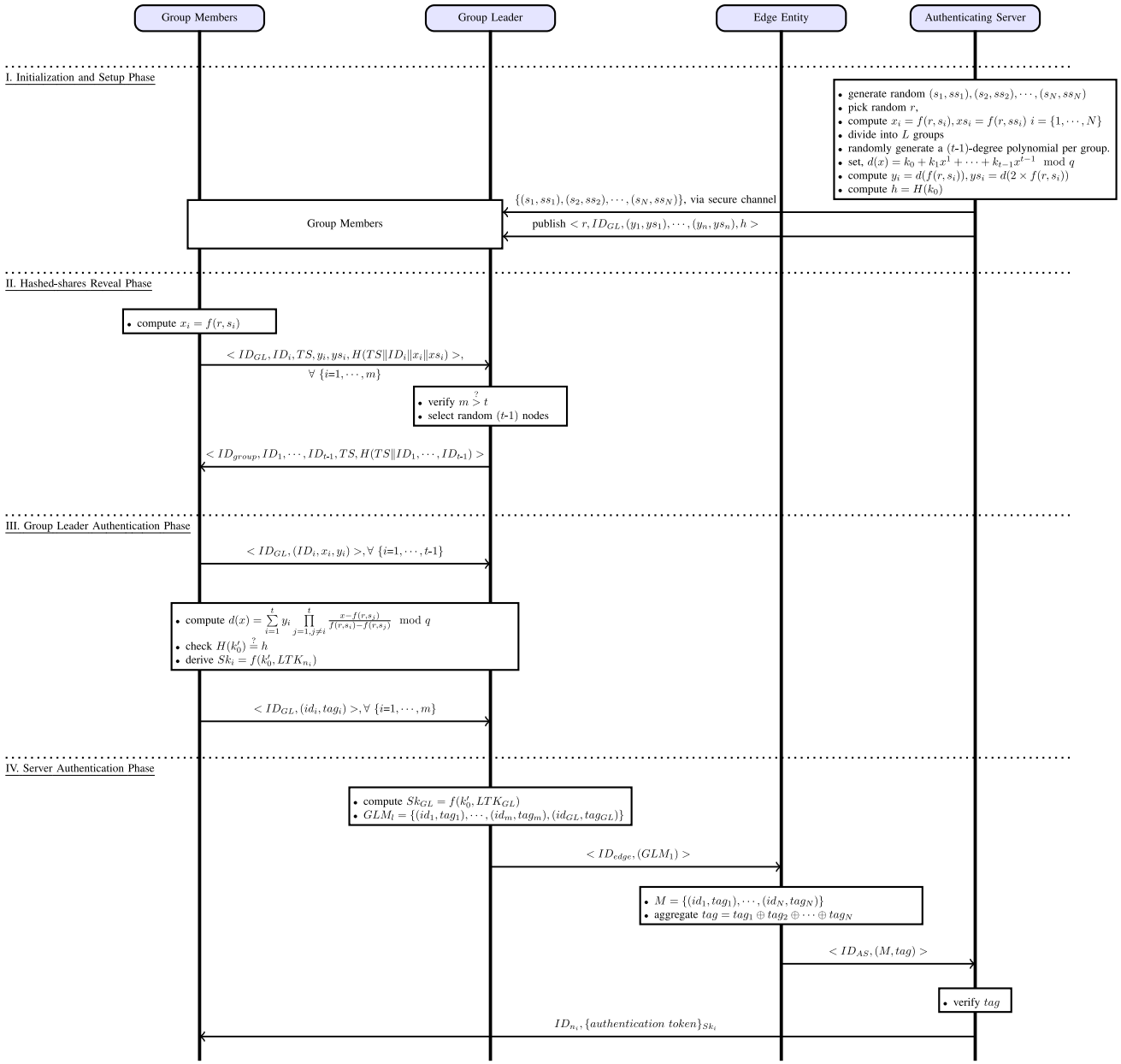

Fig. 3. Multiauthentication GAS at the edge.

D. Hashed-Shares Reveal Phase

In this phase, the m participating nodes communicate with the GL in which only $(t-1)$ members reveal one of their secret shares to the group as follows.

i. Each one of the m participants computes its tokens from the public random number r and its two stored shadow-secrets, $x_i = f(r, s_i)$ and $xs_i = f(r, ss_i)$, respectively.

ii. All m participants release the hash of their shares as follows:

$$\text{node}_i \rightarrow GL:$$
$$<\text{ID}_{\text{GL}}, \text{ID}_{\text{group}}, \text{ID}_i, \text{TS}, y_i, ys_i, H(\text{TS} \| \text{ID}_i \| x_i \| xs_i)>$$

where $\{i = 1, \ldots, m\}$. The hash is used to protect against insider impersonation attacks as discussed in Section VI.

iii. After receiving all m participants hashed shares, the GL checks the number of participants. If

a) $m < t$, the GL waits for other participants to join the authentication process for a given time window w, and if

b) $m \geq t$, the GL randomly selects $(t-1)$ nodes and requests them to reveal only one of their two shares as follows:

$$\text{GL} \rightarrow \text{ID}_{\text{group}}: <\text{ID}_1, \ldots \text{ID}_{t-1}, \text{TS}, H(\text{TS} \| \text{ID}_1, \ldots, \text{ID}_{t-1})>$$

In this phase, it is important for each node to send its hashed share in a given time window w. Specifically, the GL does not accept a participant to send its hash share after the time window w elapses. A discussion as to why this is necessary to prevent some possible attacks is provided in Section VI.

We note here that in the (n, t) threshold secret sharing schemes, there must be at least t participants to recover the secret. Thus, in our protocol, if the number of participating nodes m is less than t and the time window w elapses, the GL aborts this current authentication session and requests the AS to assign lower degree polynomial which allows lesser number of nodes to conduct the authentication protocol.

E. Group Leader Authentication Phase

In what follows, we present the steps for this phase.

i. The selected $(t-1)$ members reveal one of their (x_i, y_i) token-pairs unencrypted as follows:

$$\text{node}_i \rightarrow \text{GL}: <\text{ID}_{\text{GL}}, \text{ID}_{\text{group}}, (ID_i, x_i, y_i)>.$$

The number of revealed secrets at this point is only $(t-1)$.

ii. Using the Lagrange's interpolation formula, given in (2), the GL, and all other group nodes recover the polynomial and the secret k_0

$$d(x) = \sum_{i=1}^{t} y_i \prod_{j=1, j\neq i}^{t} \frac{x - f(r, s_j)}{f(r, s_i) - f(r, s_j)} \mod q$$
$$= k_0 + k_1 x^1 + \cdots + k_{t-1} x^{t-1} \mod q. \quad (2)$$

Here, we note that the $(t-1)$ nodes which reveal their shares recover the $(t-1)$-degree polynomial from their second shares, while the other nodes which did not reveal their shares recover the $(t-1)$-degree polynomial using one of their shares. On the other hand, an eavesdropper accesses only the $(t-1)$ shares which are not sufficient to recover the $(t-1)$-degree polynomial secret.

iii. After recovering the $(t-1)$-degree polynomial and the secret k_0, the GL and all group members are able to authenticate the $(t-1)$ members by verifying

$$H(k_0') \stackrel{?}{=} h. \quad (3)$$

We note here that if (3) holds, then, all $(t-1)$ nodes are authenticated by the GL; otherwise, the entire $(t-1)$ nodes are not authenticated. In the latter case, the GL aborts the current authentication process and requests the AS to refresh the share's random number. Selecting a new set of nodes to reveal their shares; i.e., returning back to step iii in Section V-D, causes more than $(t-1)$ nodes to reveal their shares and this exposes the $(t-1)$-degree polynomial.

iv. As for authenticating the rest of the $(m\text{-}t)$ nodes, these nodes deliver their secret shares using the recovered session key $\{x_i, x_{si}\}_{k_0}$. The GL can verify the authenticity of these shares by checking them against their hashed values as described step ii in Section V-D.

v. Each node derives its session key with the AS as follows:

$$Sk_i = f(k_0', LTK_{n_i}) \quad (4)$$

where LTK_{n_i} is the long-term key between node i and the authenticating server AS. Finally, each participating node sends to the GL its $(\text{id}_i, \text{tag}_i)$ where the tag is a standard MAC using the node's session key, $\text{tag}_i = \text{MAC}_{\text{SK}_i}(\text{id}_i)$.

F. Server Authentication Phase

After authenticating all group members, the GL follows these steps to complete the AS authentication.

vi. The GL combines all m group tags and appends its tag into, $GLM_l = \{(\text{id}_1, \text{tag}_1), \ldots, (\text{id}_m, \text{tag}_m), (\text{id}_{\text{GL}}, \text{tag}_{\text{GL}})\}$. GL sends this message to the edge entity. The key used for the GL-tag MAC is derived from the latest recovered secret, $Sk_{\text{GL}} = f(k_0', LTK_{\text{GL}})$, where LTK_{GL} is the long-term key between the GL and the AS

$$\text{GL} \rightarrow \text{Edge}: \ <\text{ID}_{\text{edge}}, \ GLM_l>.$$

vii. The edge entity collects all GLs tags and aggregates them into one tag. Specifically, M=$\{(\text{id}_1, \text{tag}_1), (\text{id}_2, \text{tag}_2), \ldots, (\text{id}_N.\text{tag}_N)\}$ and tag=$\text{tag}_1 \oplus \text{tag}_2 \oplus \cdots \oplus \text{tag}_N$.

$$\text{Edge} \rightarrow AS: \ <\text{ID}_{AS}, \ (M, \text{tag})>.$$

viii. The server receives the Agg-MAC and verifies the *tag* using the new session key for each GL.

G. Key Updates

To refresh the session keys, the AS randomly generates a new r, a new GL, and a new polynomial for each group, and publishes a new set of public values. For example, the new set for session b is

$$<r_b, \text{ID}_{\text{GLb}}, (y_{(b,1)}, ys_{(b,1)}), \ldots, (y_{(b,n)}, ys_{(b,n)}), h_b>.$$

Each node i in the group computes only the new x-tokens points using its secret-shadow shares, the new random variable, $x_i = f(r_b, s_i)$ and $xs_i = f(r_b, ss_i)$, and without the need for new secrets, and without secure channel nor redistribution nodes secret shares.

VI. Security Analysis

In what follows, we analyze the security of our protocol and prove its resilience against impersonation, reply, and asynchronous-release attacks.

A. Achieved Security Goals

We prove that our GASE scheme is a secure secret sharing scheme over $\mathbb{Z}_q$ according to Definition 1 and [16].

Theorem 1: The GASE$(\mathcal{G}, \mathcal{C})$ scheme is a secure secret sharing scheme over $\mathbb{Z}_q$ in which for every $k, k' \in \mathbb{Z}_q$, and for every $(t-1)$ subset, the distribution of $\mathcal{G}(n, t, k)$ is identical to the distribution of $\mathcal{G}(n, t, k')$.

Proof: To conduct the proof, we show that GASE is a secure sharing scheme in which the distribution of $\mathcal{G}(n, t, k)$ is identical to the distribution of $\mathcal{G}(n, t, k')$ for every $(t-1)$ subset of $\{1, \ldots, n\}$, where k, k' are two random secrets out of the key space. This implies two facts: 1) k is indistinguishable from k' for $\forall\ k, k' \in \mathbb{Z}_q$ and 2) the $(t-1)$ set of shares reveals nothing about the secret k. We prove this by the following argument. Let $\mathcal{G}(n, t, k)$ choose randomly a set $(a_1, a_2, \ldots, a_{t-1}) \stackrel{R}{\leftarrow} \mathbb{Z}_q$ such that the $(t-1)$-degree polynomial is $f(x) = k + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{t-1} x^{t-1} \in \mathbb{Z}_q[x]$ and $f(0) = k$. Then $\mathcal{G}$

chooses arbitrary $(x_1, x_2, x_3, \ldots, x_n)$ and computes their corresponding $(y_1, y_2, y_3, \ldots, y_n)$ and distributes $s_i = (x_i, y_i)$ as a secret-share for node $i = \{1, 2, 3, \ldots, n\}$. We note here that since the set $(a_1, a_2, \ldots, a_{t-1})$ is chosen uniformly over $\mathbb{Z}_q^{t-1}$, then the set $(y_1, y_2, y_3, \ldots, y_{t-1})$ is also uniformly distributed over $\mathbb{Z}_q^{t-1}$.

In what follows, we show that the algorithm $\mathcal{G}$ which sends $(a_1, a_2, a_3, \ldots, a_{t-1}) \in \mathbb{Z}_q^{t-1}$ to $(y'_1, \ldots, y'_{t-1}) \in \mathbb{Z}_q^{t-1}$ which are the y-coordinates that their corresponding x-coordinates are $(x'_1, \ldots, x'_{t-1}) \in \mathbb{Z}_q^{t-1}$ is a one-to-one map.

We prove this by a way of contradiction, suppose that the map is not a one-to-one map. This implies the existence of two distinct polynomials $d(x), p(x) \in \mathbb{Z}[x]$ of degree at most $(t-2)$ such that the polynomial $k + xd(x)$ and $k + xp(x)$ agree at the $(t-1)$ *non-zero* points of $(x'_1, \ldots, x'_{t-1})$. However, this then implies that the two polynomials $d(x)$ and $p(x)$ agree on these same $(t-1)$ which is a contradiction of the fundamentals of polynomial interpolation theorem; which specifically states that given (t) distinct-points on the x-axis plane, $(x_1, x_2, \ldots, x_t)$ and their corresponding values on the y-axis plane, $(y_1, y_2, \ldots, y_t)$, there exist only one unique $(t-1)$-degree polynomial that interpolates the data set $\{(x_1, y_1), \ldots, (x_t, y_t)\}$. Thus, given that the map is a one-to-one map, implies that the generated $(t-1)$ shares are uniformly distributed over $\mathbb{Z}_q$. It follows that the $(t-1)$ set reveals nothing about the secret k, and that k and k' are indistinguishable for $\forall\ k, k' \in \mathbb{Z}_q$. ■

Theorem 2: The GASE protocol achieves message authenticity.

Proof: We prove the authenticity of GASE by contradiction. We show that if there exists a PPT-adversary $\mathcal{A}$ that can pass the AS authentication phase, then, we can create a distinguisher $\mathcal{D}$ to break the indistinguishability of the underlying MAC primitive with non-negligible probability. We assume $\mathcal{A}$ controls the communication channel between AS and the m registered group nodes in which $\mathcal{A}$ may pick any $(\mathrm{id}_i, \mathrm{tag}_i)$ message, $1 \le i \le m$ and replace it by $(\mathrm{id}'_i, \mathrm{tag}'_i)$ which passes AS authentication with non-negligible probability, i.e., $\mathcal{A}$ succeeds in authenticating itself as a registered node. We further assume a distinguisher $\mathcal{D}$ which interacts with a challenger $\mathcal{C}$ who provides it with access to a black-box oracle $\mathcal{O}$. More precisely, $\mathcal{D}$ sends some x to $\mathcal{C}$ who flips a coin and instantiates $\mathcal{O}$ by either MAC_k where k is known only to $\mathcal{C}$, or a random function $\mathcal{R}$. Then, $\mathcal{C}$ challenges $\mathcal{D}$ with $\mathcal{O}(x)$. $\mathcal{D}$ wins the challenge if it is able to determine with non-negligible probability whether $\mathcal{O}(x) = MAC_k(x)$, or $\mathcal{O}(x) = \mathcal{R}$. In what follows, we show that if $\mathcal{A}$ exists, then $\mathcal{D}$ wins the challenge.

1) We let $\mathcal{D}$ simulate the GASE protocol for $\mathcal{A}$ where $\mathcal{D}$ runs the initialization phase, i.e., steps a–f in Section V-C for $\mathcal{A}$. Specifically, $\mathcal{D}$ selects a random $(t-1)$-degree polynomial, a random r, several m nodes, a GL. We further let the long-term key LTK_{n_i} of the ith node be known only to $\mathcal{C}$, so that it is the only entity which can evaluate SK_i.
2) $\mathcal{D}$ also runs the protocol's steps i–iii in Section V-D and steps i–v in Section V-E. In other words, at this stage of the protocol, the GL authenticates all "selected" nodes.
3) In step vi in Section V-F, $\mathcal{D}$ simulates the transmission of all nodes IDs with their associated MAC to the GL, i.e., $\{(\mathrm{id}_1, \mathrm{tag}_1), (\mathrm{id}_2, \mathrm{tag}_2), \ldots, (\mathrm{id}_m, \mathrm{tag}_m)\}$. However, for the ith node, $1 \le i \le m$, $\mathcal{D}$ queries $\mathcal{C}$ with id_i and lets $(\mathrm{id}_i, \mathrm{tag}_i) = (\mathrm{id}_i, \mathcal{O}(\mathrm{id}_i))$. $\mathcal{D}$ hopes that $\mathcal{A}$ forges the authentication message of the ith node.
4) $\mathcal{D}$ inspects the messages between $\mathcal{A}$ and AS. If $\mathcal{A}$ replaced the message $(\mathrm{id}_i, \mathrm{tag}_i)$ with $(\mathrm{id}'_i, \mathrm{tag}'_i)$ and AS authentication succeeds, then, $\mathcal{D}$ concludes that the challenge $\mathcal{O}(\mathrm{id}_i) = \mathrm{MAC}_{\mathrm{SK}_i}$; otherwise, $\mathcal{O}(\mathrm{id}_i) = \mathcal{R}$.

Let the probability of $\mathcal{A}$ forging MAC-tag be $\delta(b)$ where b is the length of the MAC-tag, then, the probability of $\mathcal{A}$ forging MAC-tag for the ith tag is given by

$$\mathsf{Pr}\Big[\mathcal{A}^{\mathrm{MAC}_{ki}}(1^b) = 1\Big] = \frac{1}{m}\delta(b).$$

Let the probability of creating a MAC-tag at random be

$$\mathsf{Pr}\Big[\mathcal{A}^{\mathcal{R}}(1^b) = 1\Big] \le \frac{1}{2^b}.$$

If we assume another MAC algorithm running a truly random function similar to $\mathcal{O}$ running $\mathcal{R}$, then, the probability of $\mathcal{A}$ forging the ith message is $(1/m)\epsilon$, where ϵ is the negligible function, because the output of $\mathcal{R}$ is uniformly distributed in $\{0, 1\}^b$ from the point of view of $\mathcal{A}$. From the construction of $\mathcal{D}$ and the probability of $\mathcal{A}$ success, it follows that:

$$\mathsf{Pr}\Big[\mathcal{D}^{\mathrm{MAC}_k}(1^b) = 1\Big] = \mathsf{Pr}[\mathcal{A}^{\mathrm{MAC}_{ki}}(1^b) = 1] = \frac{1}{m}\delta(b)$$

and

$$\mathsf{Pr}\Big[\mathcal{D}^{\mathcal{R}}(1^b) = 1\Big] = \frac{1}{m}\mathsf{Pr}\Big[\mathcal{A}^{\mathcal{R}}(1^b) = 1\Big] \le \frac{1}{m2^b}.$$

Therefore, we have

$$\Big|\mathsf{Pr}\Big[\mathcal{D}^{\mathrm{MAC}_k}(1^b) = 1\Big] - \mathsf{Pr}\Big[\mathcal{D}^{\mathcal{R}}(1^b) = 1\Big]\Big| \ge \frac{1}{m}\left(\delta(b) - \frac{1}{2^b}\right).$$

Under the semantic-security assumption that the MAC scheme is modeled using a pseudorandom function, See Definition (2), $(1/m)(\delta(b) - [1/2^b])$ must be negligible. It follows that $\delta(b)$ is negligible, which implies that $\mathcal{A}$ does not exist. ■

B. Other Attack Analysis

The following is a list of prevented attacks of our GASE protocol.

1) *Outsider Impersonation Attack:* In this attack, the outside adversary, $\mathcal{A}$, tries to impersonate a valid registered node to access its services. It is infeasible for $\mathcal{A}$ to succeed in this attack for the following reasons.
   a) There are only $(t-1)$ revealed shares in each session, and thus, an outsider $\mathcal{A}$ knows only these revealed shares. Theorem 1 proves that our GASE achieves SSS security, and $\mathcal{A}$ requires the knowledge of t shares to recover the $(t-1)$-degree polynomial. Thus, it is infeasible for $\mathcal{A}$ to create a tuple $<ID_i^{\mathcal{A}}, TS, y_i, ys_i, H(TS\|ID_i^{\mathcal{A}}\|x_i^{\mathcal{A}}\|xs_i^{\mathcal{A}})>$ that passes authentication process with a fake, $ID_i^{\mathcal{A}}$.
   b) To manipulate the list of authenticated group nodes, the attacker $\mathcal{A}$ may try to add its fake $ID_l^{\mathcal{A}}$ in the GLM MACs, $<GLM_l>$ where $GLM_l = \{(\mathrm{id}_1, \mathrm{tag}_1), \ldots, (\mathrm{id}_m, \mathrm{tag}_m), (\mathrm{id}_{\mathrm{GL}},$

$\text{tag}_{GL})\}$, described as step vi in Section V-F. This requires the knowledge of the node's session key, SK_i, and it is infeasible to brute-force the GL session key for key length ≥ 128 bits.

c) The attacker may try to access an IoT-node's i services by obtaining its session key, Sk_i. This is because all messages exchanged between the IoT-node i and the AS are encrypted using the node's session key, $Sk_i = f(k_0', LTK_{n_i})$ as shown in (4). Thus, to obtain this key, $\mathcal{A}$ needs the knowledge of k_0, LTK_i, and it is infeasible to brute-force this key for key length ≥ 128 bits.

2) *Insider Impersonation Attack:* Similar to the outsider, the inside attacker tries to impersonate another valid group member to access its services. Specifically, $\mathcal{A}$ may try to access the services for free. It is infeasible for $\mathcal{A}$ to succeed in this attack for the following reason. The inside attacker has valid shares, and thus, $\mathcal{A}$ passes the authentication process and successfully retrieves the secret k_0 as described in Section V. This enables the inside attacker to access the services. However, it is not feasible for $\mathcal{A}$ to dodge the charges, because the authenticating server AS sends the charges to the IoT-node i after the authentication process using the IoT-node i session key, $Sk_i = f(k_0', LTK_{n_i})$. Furthermore, even though $\mathcal{A}$ knows k_0, it is infeasible for $\mathcal{A}$ to derive the session key of another IoT-node i or brute force the IoT-node long-term key, LTK_{n_i}. Thus, the inside attacker $\mathcal{A}$ has the same advantages as the outside attacker in impersonating a valid registered group member.

3) *Asynchronous-Release Attack:* In the group secret sharing setting, the adversary may conduct an asynchronous-release attack in which the attacker waits for all group participants to reveal their shares, and present the adversary's good share afterword as described in Section IV-3. Our protocol is protected against this type of attack for the two types of adversaries.

a) *The Outside Attacker:* Only $(t-1)$ shares are revealed in the authentication process, see Section V-D. Thus, an outside attacker cannot recover the polynomial secrets with only $(t-1)$ shares.

b) *The Inside Attacker:* All m participants must commit their two hashed shares at the beginning of the authentication process, as seen in step ii in Section V-D; $<ID_i, TS, y_i, ys_i, H(TS\|ID_i\|x_i\|xs_i)>$. Since the GL does not accept any node to participate in the authentication process by sending its hashed shares after a given time window w, therefore, the inside attacker cannot succeed in this attack

4) *IoT-Node Collusion Attack:* The GL recovers the secret k_0 from substituting the group shares in the Lagrange's formula given in (2). For a $(t-1)$-degree polynomial, it is necessary to obtain t unique (x_i, y_i) pairs to recover k_0 using the Lagrange's formula. Because in our scheme, each IoT-node has two valid shares, thus, the inside attacker needs only $(t-2)$ other shares to recover the polynomial. On the other hand, the outside attacker needs t shares to recover the polynomial. Consequently, the inside attacker needs only $\lceil (t-2)/2 \rceil$ other colluding IoT-nodes in addition to itself, while the outside attacker needs $\lceil t/2 \rceil$ colluding IoT-nodes. It follows that our scheme resists up to $\lfloor (t-1)/2 \rfloor$ node collusion.

5) *Replay Attacks:* We use automated verification tool to formally prove that our proposed protocol is robust against these attacks.

6) *Token-Forgery Attack:* It is infeasible for an insider/outsider adversary to forge two shares to pass the authentication process for the following reason. To pass authentication, the IoT-node must commit both its two shares in a hash function as seen in step ii in Section V-D. The GL does not accept any participants to join after the time window w elapses. In Theorem 1, we prove that our GASE protocol is a secure sharing scheme. Thus, it is infeasible for an attacker to forge two shares without knowing the $(t-1)$-degree polynomial.

7) *Overtaking-IoT-Node:* If an attacker is able to physically compromise an IoT-node, the compromised device cannot be used to recover any secrets related to other devices. Once discovered, the cloud admin can revoke this device from the list so that it does not pass the AS authentication phase.

C. Analysis With Verifpal

Among formal protocol verification tools, such as ProfVerif [56], [57], [58], Tamarin [59], AVISPA [60], [61], [62], EasyCrypt [63], and Verifpal [13], the latter has a build-in SSS primitive. Thus, we find it the most suitable one to model our protocol. In this section, we present a brief background on Verifpal, and we show our protocol's implementation and verification.

Verifpal is a formal verification tool [13] which is considered a spin-off ProVerif [56], [57], [58]. Verifpal is a symbolic-model tool in which the adversary is a PPT process that runs in parallel with the protocol and has access to all system communications. To find an attack, Verifpal employs searching mechanisms that look for a trace inside the protocol which violates a specified security goal. The security goals are modeled as queries to the security properties of specific traces in the protocol. All the cryptographic build-in primitives in Verifpal are considered "perfect." The verification logic and semantics in Verifpal are based on Coq which is a formal mathematical interactive-theory-prover [64]. ProVerif semantics are also based on Coq [56], [57], [58], while AVISPA's semantics are based on Lamport's temporal logic of actions [65].

Our advantage of using Verifpal is to verify the security of our protocol utilizing the tool's build-in security primitives. Specifically, we use the Verifpal-Shamir's SSS build-in function for $(n, t) = (3, 2)$, namely, "SHAMIR_SPLIT(k)" and "SHAMIR_JOIN(s_a, s_b)," where k is the key and (s_a, s_b) are the shares, to model and verify our GASE protocol. Our Verifpal-session has three nodes and an AS, namely, NodeA, NodeB, NodeGL, and AuthS, respectively. The AS

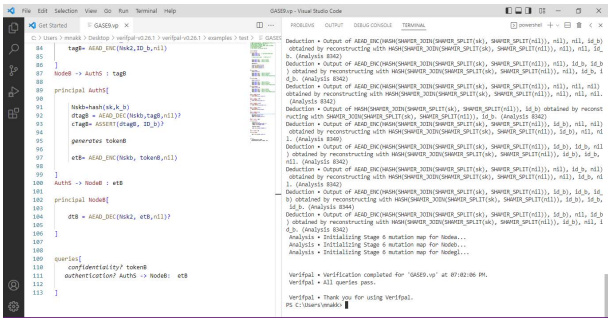

Fig. 4. Verifpal simulation results.

generates the SSS secret and distributes it securely to each group node using their long-term keys. In the authentication phase, group nodes recover SSS secret which enables each node to derive its session key and tag. Finally, the AS generates the *authentication token*, and sends it to the node. The security goals are the confidentiality and authenticity of the *authentication token* generated by the AS. Fig. 4 shows the simulation results of GASE on Verifpal.

D. Notes on Untrusted GL/IoT-Nodes Assumptions

In what follows, we present the consequences of having "untrusted" GL or IoT-nodes on the security of our GASE protocol.

1) If a GL is not a trusted entity, e.g., an adversary is able to compromise it, the only attack that can be conducted in this scenario is a denial-of-service (DoS) attack. Specifically, let us assume an untrusted GL and two types of attacks, passive or active. In both passive and active attacks, the GL cannot decrypt any secure communications between the nodes and the AS. This is because the GL does not have an access to the nodes long-term keys and cannot derive or duplicate any group node's session key even after recovering the SSS secret. Note that the node's session key is derived from the recovered SSS secret and the node's long-term key, $Sk_i = f(k_o, LTK_{n_i})$, (4). Thus, a compromised GL cannot break the confidentiality of any node's message nor impersonate any group node. A compromised GL can only drop the authentication process for group nodes, similar to DoS or dropping packet attacks, and no cryptographic technique can protect against packet dropping attacks.
2) Similarly, more than $\lfloor (t-1)/2 \rfloor$ colluding IoT-nodes can recover the SSS secret, but they cannot derive any node's session key. Consequently, these colluding nodes cannot impersonate any node nor break the confidentiality of any node's communication with the AS.
3) A field-deployed IoT-devices in the three-tier cloud-edge-IoT architecture are commonly considered as untrusted entities. This is because most of the IoT-devices are small, lightweight, and vulnerable to physical attacks. However, in our protocol, although a node-take-over physical attack mentioned in Section IV-B exposes all node's secrets to the attacker, the latter cannot access any other secrets or session keys. Specifically, in the take-over-node attack, the attacker only recovers the SSS secret and exposes the compromised node's communications, but not any other node in the network.

TABLE IV
RASPBERRY PI SIMULATION RESULTS

Operation	Symbol	Avg. execution time
Hash function[1]	T_h	$0.0482\ msec.$
HMAC function[2]	T_{MAC}	$0.0815\ msec.$
Modular multiplication[3]	$T_{mul,q}$	$9.7\ \mu sec.$
Modular addition[3]	$T_{add,q}$	$6.7\ \mu sec.$
Symmetric-key encryption/decryption[4]	T_{enc}	$0.043\ msec.$
ECC multiplication[5]	EC_{mul}	$0.38\ msec.$
ECC addition[5]	EC_{add}	$0.089\ msec.$

[1]: SHA-256 with data size= 1024 Bytes.
[2]: HMAC-SHA25 with data size= 1024 Bytes.
[3]: Modulus size= 256 bits
[4]: AES-128-CBC with data size= 1024 Bytes and key-size= 128-bits.
[5]: Barreto-Naehrig Curve P-256.

VII. COMPARATIVE EVALUATION

Similar to [66], we ran the following operations, $T_{\text{mul},q}$,$T_{\text{add},q}$, ECC_{mul}, ECC_{add}, HMAC with SHA-256, and they are modular multiplication, modular addition, ECC multiplication, ECC addition, HMAC, Hash, respectively, on Raspberry Pi 4 Model B/8GB/Broadcom-BCM2711, Quad-core Cortex-A72-1.5GHz (ARM v8) 64-bit SoC processor, Python-library pblib, [67]. Table IV shows the operations simulated on Raspberry Pi-Model 4. We note here that we presented in Section II several group authentication protocols; however, in the section, we compare the most relevant schemes that are based on SSS with our scheme, specifically [6], [7], [8], [9], and present it in Table V. We follow the theoretical analysis and notations given by Chein [9] to determine the computational complexity for users and the GL in our GASE scheme. Unlike other schemes, we require modular addition and multiplication operations in $GF(q)$. On the other hand, Harn's scheme-3 uses addition and multiplication operations in modular field $GF(q)$ as well as computing the modular exponentiation in another modular field, $GF(p)$ [6]. Also, Aydin *et al.* use $GF(q)$ operations as well as ECC multiplications and additions for authentication and key agreement.

Computational Complexity: For m participants in the authentication process, each user computes the shares, $x_i = f(r, s_i)$ and $xs_i = f(r, ss_i)$ and sends them to the GL for authentication. We assume that the two-variable one-way function is equivalent $\cong$ 1 hash operation. For authentication, the GL computes Lagrange's formula given in (2). The GL and all other members compute, $= 2 \times \text{hash} + \left[(2(m-2)+2) \times T_{\text{mul},q} + 1 \times T_{inv,q} \right] + (m-1) \times T_{\text{add},q}$, where $T_{\text{mul},q}$, $T_{inv,q}$, and $T_{\text{add},q}$ are multiplication, multiplicative inverse, and addition in q, $GF(q)$, respectively. Finally, the GL authenticates the recovered secrets by performing $(m+1) \times$ hash for hashing and verifying nodes original shares and *ID*s, see (3) and

TABLE V
PERFORMANCE COMPARISONS GROUP AUTHENTICATION PHASE

Ref	Comm/user	Shares/user	Computations/user	Complexity/ET♦	Key-agreement/ user
Proposed	$(t+m)$	2 *shadows*	$[(2(m-2)+2)\times T_{mul,q}+1\times T_{inv,q}]+(m-1)\times T_{add,q}+(m+2)\times hash$	$\cong (2m+238)T_{mul,q} \cong 2.7\ msec$ ❖	$=2\times hash \cong 5.8\mu sec$
Harn-3✳ [6]	$2\times(m$-1)	2 *shares*	$2[(2(m-2)+3)\times T_{mul,q}+1\times T_{inv,q}]+1\times T_{exp,p}+(m-1)\times mul_p+1\times hash$	$\cong (45m+1418)T_{mul,q} \cong 22\ msec$	NA ⋆
Aydin-1 [7]	$(m$-1)	1 *share*	$1\times EC_{mul}$	$\cong (1189)T_{mul,q} \cong 12\ msec$★	$(m$-1$)\times ECDH+[(2(m-2)+2)\times T_{mul,q}+T_{inv,q}+(m-1)\times T_{add,q}+1\times hash] \cong 17.572\ msec$
Aydin-2 [7]	$(m$-1)	1 *share*	$[(2(m-2)+1)\times T_{mul,q}+1\times T_{inv,q}]+1\times EC_{mul}+(m-1)\times EC_{add}$	$\cong (7m+1421)T_{mul,q} \cong 15\ msec$ ✪	$=(m-1)\times ECDH+[(2(m-2)+2)\times T_{mul,q}+T_{inv,q}+(m-1)\times T_{add,q}+1\times hash] \cong 17.572\ msec$
Li [8]	$(m$-1)	θ *shares*	$\theta[(2(m-2)+3)\times T_{mul,q}+1\times T_{inv,q}]+(m-1)\times T_{add,q}+1\times hash$	$\cong 41\theta(2m+239)T_{mul,q} \cong 222\ msec$	$1\times EC_{mul}+1\times hash+1\times MAC \cong 0.78\ msec$
Chien [9]	$(m$-1)	1 *share*	$[(2(m-1)+2)\times T_{mul,q}+1\times T_{inv,q}]+1\times EC_{mul}+m\times EC_{add}+2\times T_{pair}$	$\cong (7m+6785)T_{mul,q} \cong 67\ msec$	NA ⋆

❖: For case of GL authenticating rest of $(m$-$t)$ nodes, we have extra encryption. Specifically, $ET \cong [(2(m-2)+2)\times T_{mul,q}+1\times T_{inv,q}]+(m-1)\times T_{add,q}+(m+2)\times hash+(m$-$t)\times T_{enc} \cong (2m+238)T_{mul,q}+(m$-$t)\times T_{enc} \cong 2.71\ msec$
✳: Harn's scheme-3 [6].
★: Aydin case 1: Group manager confirming group members [7].
✪: Aydin case 2: Any group member confirming other group members [7].
♦: Based on Chien's approximation [9]; parameters are $t=10$, $m=20$, $\theta=2$.
♦: All execution times listed in Table IV are scaled to 32-byte data blocks.
⋆: not available in the scheme.

step ii in Section V-D. For a fair comparison with [7], [8], and [9] we follow Chien's approximation of ignoring hash and field-addition operations, and considering $T_{inv,q} \cong 240T_{\text{mul},q}$. Thus, our approximated time complexity for the GL is $\cong (2m-4+2+240)\times T_{\text{mul},q} \cong (2m+238)T_{\text{mul},q}$.

On the other hand, the Aydin's scheme has two different authentication processes, one with group manager confirming all group members, and one with any group member confirming other members, cases 1 and 2 in Table V, respectively. In the first case, each user computes only one ECC multiplication, specifically $f(x_i)\times P$, and sends it to the group manager for authentication. Thus, based on Chien's approximation, the complexity per user for Aydin case 1 is $=1\times EC_{\text{mul}} \cong 29T_{\text{mul},P} \cong 29(41)T_{\text{mul},q} \cong (1189)T_{\text{mul},q}$. In Aydin case 2, the authenticating node in the group computes all shares from each other node and verifies it with the public quantity $Q=sP$. Thus, the authenticating node must perform elliptic curve field multiplications and additions, and the approximated time is $=\left[(2(m-2)+1)\times T_{\text{mul},q}+1\times T_{inv,q}\right]+1\times EC_{\text{mul}}+(m-1)\times EC_{\text{add}} \cong (7m+1421)T_{\text{mul},q}$, where EC_{mul} and EC_{add} are the ECC multiplication and addition, respectively. Note that Aydin case 2 is the generic case where any node can be GL, and hence, it is the case that is closely related to our protocol.

Table V shows comparisons between our scheme and other related schemes, [6], [7], [8], [9] in terms of the computation complexity per user, approximated complexity per user, and group key agreement complexities. In Fig. 5, on the other hand, shows the approximated number of multiplications per number of group users m. We note here that we selected the range of group size from 1–350 for the sake of comparisons with other schemes; specifically, the Adyin's scheme [7] have the group size 1–10, the Li's scheme [8] have the group size 1–180, and the Chien's scheme [9] have the group size 1–350. Nevertheless, our scheme can support larger group sizes. Our

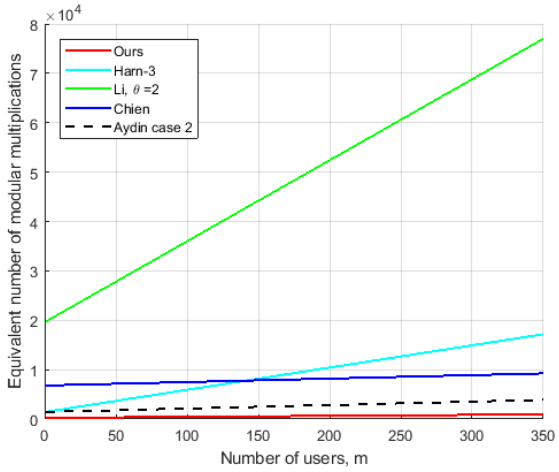

Fig. 5. Number of multiplications per user for different schemes.

scheme has the least computations per user. Furthermore, our node key agreement requires the least amount of computations when compared to others; Specifically, the Aydin scheme requires $=(m-1)\times ECDH+\left[(2(m-2)+2)\times T_{\text{mul},q}+T_{inv,q}+(m-1)\times T_{\text{add},q}+1\times \text{hash}\right]$ per user and also the Li's scheme requires $=1\times EC_{\text{mul}}+1\times \text{hash}+1\times MAC$ per user.

Communication Per User: Unlike other schemes, we refresh secrets in each session. The AS publishes a new set of public values, $(r, \text{ID}_{\text{GL}}, (y_1, ys_1), (y_2, ys_2), \ldots, (y_n, ys_n), h)$, in each session. Thus, assuming each parameter is 32 Bytes, then, in each session, the AS publishes $=32\times(2n+2)$ Bytes. For authentication, there are three packets exchanged in the group; specifically, $Pkt_1 = < ID_i, TS, y_i, ys_i, H(TS\|ID_i\|x_i\|xs_i) >$, $Pkt_2 = < \text{ID}_1, \ldots, \text{ID}_{t-1}, TS, H(TS\|\text{ID}_1, \ldots, \text{ID}_{t-1}) >$, and $Pkt_3 = < (ID_i, x_i, y_i) >$. Assuming $TS=4$ Bytes, $ID=3$ Bytes, SHA-256 $=32$ Bytes, $x_i=32$ Bytes, and $y_i=$

32 Bytes, the total communication overhead per group is $((m + 2t - 2) \times 3$ Bytes $+3 \times 4$ Bytes $+2t \times 32$ Bytes $+ (m+1) \times 32$ Bytes $= (35m + 70t + 18)$ Bytes. The communication overhead per user is $m + t$. Other schemes [6], [7], [8], [9], on the other hand, require less communication, because they do not support any key update mechanism.

Number of Shares Per Node: In our scheme, we require two secret-shadows per node, say 2×32 Bytes. All other schemes require each group node to store only one secret-share, except for Harn's scheme-3, where each node is required to store two secret shares, each corresponding to a different polynomial. Similarly, the Li's scheme also requires for each node to store θ-polynomials secrets.

VIII. Conclusion

We proposed a lightweight GAS suitable for EC paradigm and massive node authentications. The protocol is based on Yang's multisecret sharing scheme and Katz's Agg-MAC. The scheme provides multiple authentication and key agreement to a group of nodes in the three-tier cloud-edge-IoT framework. In addition, our scheme allows for session key refreshing and provides forward secrecy. We presented the security analysis of our scheme which includes a proof of the security properties and a discussion of prevented attacks. We compared our GASE protocol to other recent proposals and showed that it has lesser computational complexity than others.

References

[1] A. Holst. "Number of IoT connected devices worldwide 2019–2030." 2021. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide

[2] 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.

[3] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, and S. A. Malik, "Fog/edge computing-based IoT (FECIoT): Architecture, applications, and research issues," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4118–4149, Jun. 2019.

[4] A. Alwarafy, K. A. Al-Thelaya, M. Abdallah, J. Schneider, and M. Hamdi, "A survey on security and privacy issues in edge-computing-assisted Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4004–4022, Mar. 2021.

[5] 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.

[6] L. Harn, "Group authentication," *IEEE Trans. Comput.*, vol. 62, no. 9, pp. 1893–1898, Sep. 2013.

[7] Y. Aydin, G. K. Kurt, E. Ozdemir, and H. Yanikomeroglu, "A flexible and lightweight group authentication scheme," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10277–10287, Oct. 2020.

[8] J. Li, M. Wen, and T. Zhang, "Group-based authentication and key agreement with dynamic policy updating for MTC in LTE-A networks," *IEEE Internet Things J.*, vol. 3, no. 3, pp. 408–417, Jun. 2016.

[9] H.-Y. Chien, "Group authentication with multiple trials and multiple authentications," *Security Commun. Netw.*, vol. 2017, May 2017, Art. no. 3109624.

[10] J. Katz and A. Y. Lindell, "Aggregate message authentication codes," in *Proc. Cryptograph. Track RSA Conf.*, 2008, pp. 155–169.

[11] C. Lai, H. Li, R. Lu, R. Jiang, and X. Shen, "LGTH: A lightweight group authentication protocol for machine-type communication in LTE networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2013, pp. 832–837.

[12] C. Lai, R. Lu, D. Zheng, H. Li, and X. S. Shen, "GLARM: Group-based lightweight authentication scheme for resource-constrained machine to machine communications," *Comput. Netw.*, vol. 99, pp. 66–81, Apr. 2016.

[13] N. Kobeissi, G. Nicolas, and M. Tiwari, "Verifpal: Cryptographic protocol analysis for the real world," in *Proc. Int. Conf. Cryptol. India*, 2020, pp. 151–202.

[14] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[15] G. R. Blakley, "Safeguarding cryptographic keys," in *Proc. Int. Workshop Manag. Requirements Knowl.*, 1979, p. 313.

[16] D. Boneh and V. Shoup, "A graduate course in applied cryptography," IETF, Internet-Draft draft 0.5, Jan. 2020.

[17] P. Shabisha, A. Braeken, P. Kumar, and K. Steenhaut, "Fog-orchestrated and server-controlled anonymous group authentication and key agreement," *IEEE Access*, vol. 7, pp. 150247–150261, 2019.

[18] K. Kaya and A. A. Selçuk, "Threshold cryptography based on Asmuth-Bloom secret sharing," *Inf. Sci.*, vol. 177, no. 19, pp. 4148–4160, 2007.

[19] A. Yang, J. Weng, K. Yang, C. Huang, and X. Shen, "Delegating authentication to edge: A decentralized authentication architecture for vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1284–1298, Feb. 2022.

[20] Y. Qiu and M. Ma, "Secure group mobility support for 6LoWPAN networks," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1131–1141, Apr. 2018.

[21] H. Yildiz, M. Cenk, and E. Onur, "PLGAKD: A PUF-based lightweight group authentication and key distribution protocol," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5682–5696, Apr. 2021.

[22] X. Ren, J. Cao, M. Ma, H. Li, and Y. Zhang, "A novel PUF-based group authentication and data transmission scheme for NB-IoT in 3GPP 5G networks," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3642–3656, Mar. 2022.

[23] M. Huang, B. Yu, and S. Li, "PUF-assisted group key distribution scheme for software-defined wireless sensor networks," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 404–407, Feb. 2018.

[24] P. Dong, W. Wang, X. Shi, and T. Qin, "Lightweight key management for group communication in body area networks through physical unclonable functions," in *Proc. IEEE/ACM Int. Conf. Connected Health Appl. Syst. Eng. Technol. (CHASE)*, 2017, pp. 102–107.

[25] S. Chen, B. Li, Z. Chen, Y. Zhang, C. Wang, and C. Tao, "Novel strong-PUF-based authentication protocols leveraging Shamir's secret sharing," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14408–14425, Aug. 2022.

[26] S. Li, I. Doh, and K. Chae, "A group authentication scheme based on Lagrange interpolation polynomial," in *Proc. 10th Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput. (IMIS)*, 2016, pp. 386–391.

[27] Y. Guo, Z. Zhang, and Y. Guo, "Fog-centric authenticated key agreement scheme without trusted parties," *IEEE Syst. J.*, vol. 15, no. 4, pp. 5057–5066, Dec. 2021.

[28] Q. Cheng, C. Hsu, Z. Xia, and L. Harn, "Fast multivariate-polynomial-based membership authentication and key establishment for secure group communications in WSN," *IEEE Access*, vol. 8, pp. 71833–71839, 2020.

[29] J. Cao, P. Yu, M. Ma, and W. Gao, "Fast authentication and data transfer scheme for massive NB-IoT devices in 3GPP 5G network," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1561–1575, Apr. 2019.

[30] C. Lai, H. Li, R. Lu, and X. S. Shen, "SE-AKA: A secure and efficient group authentication and key agreement protocol for LTE networks," *Comput. Netw.*, vol. 57, no. 17, pp. 3492–3510, 2013.

[31] S. Basudan, "LEGA: A lightweight and efficient group authentication protocol for massive machine type communication in 5G networks," *J. Commun. Inf. Netw.*, vol. 5, no. 4, pp. 457–466, Dec. 2020.

[32] J. Cao, Z. Yan, R. Ma, Y. Zhang, Y. Fu, and H. Li, "LSAA: A lightweight and secure access authentication scheme for both UE and mMTC devices in 5G networks," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5329–5344, Jun. 2020.

[33] H.-Y. Chien, "Group-oriented range-bound key agreement for Internet of Things scenarios," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1890–1903, Jun. 2018.

[34] D. Chaum and E. Van Heyst, "Group signatures," in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, 1991, pp. 257–265.

[35] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2003, pp. 416–432.

[36] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. Annu. Int. Cryptol. Conf.*, 2004, pp. 41–55.

[37] C. Asmuth and J. Bloom, "A modular approach to key safeguarding," *IEEE Trans. Inf. Theory*, vol. TIT-29, no. 2, pp. 208–210, Mar. 1983.

[38] S. Iftene, "General secret sharing based on the Chinese remainder theorem with applications in E-voting," *Electron. Notes Theor. Comput. Sci.*, vol. 186, pp. 67–84, Jul. 2007.

[39] J. Cui, X. Chen, J. Zhang, Q. Zhang, and H. Zhong, "Toward achieving fine-grained access control of data in connected and autonomous vehicles," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 7925–7937, May 2001.
[40] M. H. Ibrahim, "Octopus: An edge-fog mutual authentication scheme," *Int. J. Netw. Security*, vol. 18, no. 6, pp. 1089–1101, 2016.
[41] 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.
[42] 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.
[43] S. Brands and D. Chaum, "Distance-bounding protocols," in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, Berlin, Germany, 1993, pp. 344–359.
[44] D. Singelee and B. Preneel, "Location verification using secure distance bounding protocols," in *Proc. IEEE Int. Conf. Mobile Adhoc Sens. Syst. Conf.*, 2005, p. 840.
[45] Y.-J. Tu and S. Piramuthu, "RFID distance bounding protocols," in *Proc. 1st Int. EURASIP Workshop RFID Technol.*, 2007, pp. 67–68.
[46] G. P. Hancke and M. G. Kuhn, "An RFID distance bounding protocol," in *Proc. 1st Int. Conf. Security Privacy Emerg. Areas Commun. Netw. (SECURECOMM)*, 2005, pp. 67–73.
[47] G. Kapoor, W. Zhou, and S. Piramuthu, "Distance bounding protocol for multiple RFID tag authentication," in *Proc. IEEE/IFIP Int. Conf. Embedded Ubiquitous Comput.*, vol. 2, 2008, pp. 115–120.
[48] J. He and E. Dawson, "Multistage secret sharing based on one-way function," *Electron. Lett.*, vol. 30, no. 19, pp. 1591–1592, 1994.
[49] C.-C. Yang, T.-Y. Chang, and M.-S. Hwang, "A (t, n) multi-secret sharing scheme," *Appl. Math. Comput.*, vol. 151, no. 2, pp. 483–490, 2004.
[50] L.-J. Pang and Y.-M. Wang, "A new (t, n) multi-secret sharing scheme based on Shamir's secret sharing," *Appl. Math. Comput.*, vol. 167, no. 2, pp. 840–848, 2005.
[51] H.-Y. Chien, J.-K. Jan, and Y.-M. Tseng, "A practical (t, n) multi-secret sharing scheme," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 83, no. 12, pp. 2762–2765, 2000.
[52] J. Zhao, J. Zhang, and R. Zhao, "A practical verifiable multi-secret sharing scheme," *Comput. Stand. Interfaces*, vol. 29, no. 1, pp. 138–141, 2007.
[53] J. Shao and Z. Cao, "A new efficient (t, n) verifiable multi-secret sharing (VMSS) based on YCH scheme," *Appl. Math. Comput.*, vol. 168, no. 1, pp. 135–140, 2005.
[54] K. Kifayat, M. Merabti, Q. Shi, and D. Llewellyn-Jones, "An efficient multi-parameter group leader selection scheme for wireless sensor networks," in *Proc. Int. Conf. Netw. Services Security*, 2009, pp. 1–5.
[55] C. Boyd, A. Mathuria, and D. Stebila, *Protocols for Authentication and Key Establishment*, vol. 1. Heidelberg, Germany: Springer, 2003.
[56] B. Blanchet, "Security protocol verification: Symbolic and computational models," in *Proc. Int. Conf. Principles Security Trust*, Berlin, Germany, 2012, pp. 3–29.
[57] B. Blanchet, "Automatic verification of security protocols in the symbolic model: The verifier ProVerif," in *Foundations of Security Analysis and Design VII*. Cham, Switzerland: Springer, 2013, pp. 54–87.
[58] B. Blanchet, *Modeling and Verifying Security Protocols With the Applied Pi Calculus and ProVerif*, vol. 1. Hanover, MA, USA: Now Publ., Oct. 2016, pp. 1–135.
[59] P. Remlein, M. Rogacki, and U. Stachowiak, "Tamarin software—the tool for protocols verification security," in *Proc. Baltic URSI Symp. (URSI)*, 2020, pp. 118–123.
[60] "Automated validation of Internet security protocols and applications." AVISPA. Accessed: 2021. [Online]. Available: http://www.avispa-project.org/
[61] "SPAN, the security protocol ANimator for AVISPA." SPAN + AVISPA. Accessed: 2021. [Online]. Available: http://people.irisa.fr/Thomas.Genet/span/
[62] D. Von Oheimb, "The high-level protocol specification language HLPSL developed in the EU project AVISPA," in *Proc. APPSEM Workshop*, 2005, pp. 1–17.
[63] C. Baritel-Ruet. "Formal security proofs of cryptographic: A necessity achieved using EasyCrypt." Accessed: 2022. [Online]. Available: https://tel.archives-ouvertes.fr/tel-03177617/document
[64] B. Barras *et al.*, "The Coq proof assistant reference manual : Version 6.1," Ph.D. dissertation, Dept. Comput. Sci., Inria, Paris, France, 1997.
[65] L. Lamport, "The temporal logic of actions," *ACM Trans. Program. Lang. Syst.*, vol. 16, no. 3, pp. 872–923, 1994.
[66] 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.
[67] "Raspberry Pi." Accessed: Apr. 2022. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

Mouna Nakkar (Member, IEEE) received the M.A.Sc. degree in information systems security from the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC, Canada, in 2018, where she is currently pursuing the Ph.D. degree in information and systems engineering.

Her current research interests include edge computing networks, security in Internet of Things, and post-quantum cryptography.

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

She was an NSERC Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, and an NSERC Canada Graduate Scholar with Concordia University. She is currently an Assistant Professor and a Graduate Program Director with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, 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, Cairo; and Cairo University. He also served on more than 60 technical program committees of cryptography and data security conferences. His research interests include cryptology, malware analysis, and cyber–physical systems security. He has more than 230 referred journal and conference publications in areas related to his research interests.

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