Lightweight Authentication for Edge Computing

Mouna Nakkar

A Thesis

in

The Concordia Institute

for

Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy (Information and Systems Engineering) at
Concordia University
Montréal, Québec, Canada

December 2024

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mouna Nakkar**

Entitled: **Lightweight Authentication for Edge Computing**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Ahmed Soliman

_____ External Examiner
Dr. Nora Boulahia-Cuppens

_____ External to Program
Dr. Walaa Hamouda

_____ Examiner
Dr. Jamal Bentahar

_____ Examiner
Dr. Abdessamad Ben Hamza

_____ Supervisor
Dr. Amr M. Youssef

_____ Co-supervisor
Dr. Riham AlTawy

Approved by _____
Farnoosh Naderkhani, Graduate Program Director

December 4^{th}, 2024
Date of Defence

_____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Lightweight Authentication for Edge Computing

Mouna Nakkar, Ph.D.

Concordia University, 2024

Edge computing (EC) is one of the most promising decentralized network paradigms in the proliferation era of the Internet of Things (IoT). Although this paradigm has high potential in terms of performance and de-centralization, it carries several security concerns. One of the most important security properties for the EC paradigm is authenticity. It allows different edge computing entities to verify each other through cryptographic means. Additionally, authenticity regulates access control to different edge computing resources and data. There are several types of authentications, one-way authentication, mutual authentication, broadcast authentication, group authentication, and others. In our thesis, we focus on designing security protocols for different edge computing applications that require either mutual authentication, broadcast authentication, or group authentication. In all our proposed protocols we utilize lightweight security primitives suitable for the three-tier cloud-edge-IoT architecture.

In the first protocol, we utilize lightweight cryptographic primitives to design mutual authentication for EC broadcast messages. Specifically, the used primitives are only hash-based one-way chain, symmetric-key cryptography, and a hash function. The protocol establishes key-agreement for group and individual nodes in each session. The achieved security properties for our protocol are mutual-authentication for broadcast messages,

message secrecy, message integrity, and forward secrecy. We formally define and prove the main security properties of our protocol theoretically using the indistinguishablity game. We compare our protocol to other lightweight protocols in terms of security and performance to prove its advantages in terms of computations, communication overhead, and storage.

Motivated by the fact that mass authentication is one of the desirable security features in the edge computing paradigm, our second proposed protocol is a lightweight group authentication scheme (GAS) with session key-agreement. The protocol utilizes lightweight cryptographic primitives, namely, Shamir's secret sharing (SSS) scheme and aggregated message authentication code (Aggregated-MAC). Unlike other group authentication schemes, our protocol provides multiple asynchronous authentications. Furthermore, we implement a simple key refreshing mechanism such that in each session, a new session-key between group nodes and the authenticating server is established without the need for redistributing new shares. Our security analysis includes proving that our protocol provides group authenticity, message forward secrecy, and prevents several attacks.

Extending our group authentication design, our third security protocol is a flexible GAS based on Physical Unclonable Function (PUF) and Shamir's secret sharing scheme. Specifically, we apply PUFs on SSS and utilize the SSS-homomorphic property to achieve multiple-time group authentications with the same set of shares. Our scheme is lightweight, establishes a new group key-agreement per session, and supports efficient node-evicting mechanism. Furthermore, in our protocol, the group nodes do not store any shares; instead, the nodes derive their secret-shares from their PUF-responses. We formally analyze our protocol theoretically and with automated tools, Automated Validation of Internet Security Protocol and Applications (AVISPA), to prove that our scheme achieves message secrecy and authenticity.

Finally, we propose a lightweight symmetric-key based protocol which provides edge-IoT mutual authentication, forward secrecy, backward secrecy, and anonymity. The security primitives used in our fourth protocol are pseudo-random function (PRF), random number generation, a hash function, and xor. We prove the security goals of the protocol and compare it to other lightweight authentication protocols.

Acknowledgments

I would like to take this opportunity to thank all the professional helping hands who contributed towards finishing this dissertation. First, I would like to express my sincere gratitude and appreciation towards my supervisors, Professor Amr Youssef and Dr. Riham AlTawy. I would like thank Prof. Youssef for his mentoring, guidance, continuous technical feedback, patience, encouragement, and immense knowledge. I would like to thank Dr. Riham for her technical feedback, discussions, and help. I have learnt a lot from my supervisors, and I appreciate their mentoring, guidance, and support.

I would like to sincerely thank the examining committee: Dr. Walaa Hamouda, Dr. Jamal Bentahar, and Dr. Abdessamad Ben Hamza for their helpful insights, valuable discussion, and guidance throughout each milestone of the Ph.D. journey.

I would like to also thank the Concordia Institute for Information Systems Engineering, its faculty, and the graduate school for all their help and support in finishing this dissertation. I would like to also thank my colleague Mohammed Mahdy for the technical discussions, feedback, and help. Finally, I would like to thank all those who made the completion of this thesis possible.

Mouna Nakkar

Table of Contents

List of Figures

List of Tables

List of Acronyms

3GPP	Third Generation Partnership Project
AP	Access Point
AES	Advanced Encryption Standard
AMI	Advanced Metering Infrastructure
Aggregated-MAC	Aggregate message authentication code
AI	Artificial Intelligence
AE	Authenticated Encryption
AKE	Authentication and Key-agreement
AVISPA	Automated Validation of Internet Security Protocol and Applications
BAN Logic	Burrows–Abadi–Needham Logic
BAN	Building Area Network
CCTV	Closed-circuit Television
CK	Canetti-Krawczyk
CRP	Challenge Response Pair
CRT	Chinese Reminder Theorem
CC	Cloud Computing
CIA	Confidentiality Integrity Availability
CL-AtSe	Constraint Logic-based ATtack SEarcher
CDN	Content Delivery Network
CPS	Cyber-Physical System

DDH	Decisional Diffie Hellman
DP	Differential Privacy
DHKE	Diffie-Hellman Key exchange
DL	Discrete Logarithm
DoS	Denial-of-Service attacks
DDoS	Distrusted Denial-of-Service attacks
DY	Dolev-Yao
EC	Edge Computing
EI	Edge Intelligence
ECC	Elliptic Curve Cryptography
ECC	Error Correction Code
ETSI	European Telecommunications Standards Institute
FC	Fog Computing
FEC	Fog/Edge Computing
FE	Fuzzy Extractors
GW	Gate Way
GAS	Group Authentication Scheme
HMAC	Hash-based message authentication code
HLPSL	High Level Protocol Specification Language
HAN	Home Area Network
HE	Homomorphic Encryption
IIoT	Industrial Internet of Things
ISG	Industry Specification Group
IT	Information Technology
IC	Integrated Circuit
ITS	Intelligent Transportation System
IoT	Internet of Things

IoV	Internet of Vehicles
IPsec	Internet Protocol Security
ISP	Internet Service Provider
LP	Location Privacy
LTE-A	Long-Term Evolutionary-Advanced
MiTM	Man-in-The-Middle
mMTC	Massive machine type communication
MAC	Message Authentication Code
MQTT	Message Queue Telemetry Transport
MCC	Mobile Cloud Computing
MEC	Mobile Edge Computing
NAN	Neighborhood Area Network
OFMC	On-the-Fly Model Checker
OTS	One-Time-Signature
PUF	Physical Unclonable Function
PPT	Probabilistic-polynomial-time
PRFs	Pseudo Random Functions
PKC	Public Key Cryptography
RFID	Radio frequency identification
RoR	Real-or-Random model
rFE	(reverse) fuzzy extractors
RSA	Rivest-Shamir-Adleman
SSS	Shamir's Secret Sharing
SATMC	SAT-based Model Checker
SHA	Secure Hash Algorithm
SoG	Sequence-of-games
SPAN	Security Protocol ANimnator

SCA	Side Channel Attack
TLS	Transport Layer Security
VSS	Verifiable Secret Sharing
V2G	Vehicle to Grid
VC	verifiable computing
VM	virtual machines
WSN	Wireless Sensor Network

Chapter 1

Introduction

1.1 Overview

One of the most prominent de-centralized paradigms is edge computing (EC). EC offers abundance of services and advantages such as, efficiency, mobility, scalability of supporting large number of nodes, computation offloading, data streaming, enhancing wireless access, and others. With the migration into the de-centralized Internet-of-Things (IoT) smart applications and the accelerated adoption of the edge-computing paradigm, security remains a big concern. Indeed, new security issues are arising such as user's privacy, location privacy, anonymity, untraceability, and confidentiality due to the increasing dependency on sensors, cameras, and smart phones in the EC smart applications. On the other hand, the limited resources of the edge devices render the implementations of traditional cryptography solutions useless.

1.2 Motivation

Authentication is one of the major security goals in edge computing. However, designing authentication solutions for the EC-paradigm is more challenging than the typical centralized cloud computing. This is because authenticating nodes from delegated

EC severs increases the attack surface from the typical cloud-node authentication. Also, most of the edge-computing authentication solutions proposed in the literature are heavy-weight, require large storage, or consume large communication bandwidth. Additionally, most authentication solutions do not support multiple-nodes authentications or mass authentication; instead, the edge server authenticates one IoT node at a time. This puts a burden on the edge server and may jeopardize security.

In our research, we study the security of the edge computing paradigm and propose lightweight authentication and key-agreement solutions. We propose several solutions for mass authentication in which the edge server authenticates a large number of nodes in one authentication step. Also, we propose a lightweight authentication solution for message broadcast EC applications. In our proposed security protocols, we also focus on the efficiency and performance of the schemes. Specifically, most of our solutions utilize lightweight cryptographic primitives such as symmetric-key encryption, Shamir's secret sharing (SSS), Physical Unclonable Function (PUF), hash function, hashed-based message authentication code (HMAC), one-way hash chain, Pseudo-random Function (PRF), and the xor gate. For group and mass authentication, we propose efficient share redistributions and updates.

1.3 Contributions

We study the security of the edge computing paradigm. Specifically, our security goals are focused on providing authentication, group authentications, and key-agreement for edge computing applications in the three-tier cloud-edge-IoT architecture. Because of the low-end IoT devices, we utilize different lightweight cryptography primitives. The following summarize our contributions:

- We propose a lightweight authentication scheme for the edge computing applications that requires messaging broadcast [126]. The protocol utilizes lightweight primitives

such as symmetric encryption, hash function, and one-way hash-chain. The semantic security of the protocol is formally proven using theoretical indistinguishable game analysis.

- We propose a group authentication scheme (GAS) for edge computing applications [128]. The scheme utilizes Shamir's secret sharing scheme, Yang's secret-sharing [191], Aggregated-MAC [83], and symmetric encryption to providing mass authentication for a group of an IoT nodes, and key-agreement for the individual edge nodes. The security protocol is formally proven using theoretical and automated analysis.

- We extend our work of designing group authentication schemes utilizing physical security primitive, namely, PUFs [129]. The protocol provides group authentication for a group of IoT nodes and individual IoT node key-agreement. The protocol uses the node's PUF responses as the shares for Shamir's secret sharing. The security protocol is formally proven using theoretical and automated analysis.

- We propose a lightweight authentication and key-agreement protocol for the edge computing paradigm that provides forward and backward secrecy [127].

Other research works conducted during the tenure of this Ph.D. have been published in [156].

1.4 Thesis Outline

- Chapter 2 provides background on the edge computing paradigm architecture and security.

- Chapter 3 presents lightweight broadcast authentication protocol for the edge computing applications.

- Chapter 4 presents lightweight group authentication scheme using Shamir's Secret Sharing.
- Chapter 5 presents a lightweight group authentication scheme leveraging Shamir's Secret Sharing and physical unclonable devices.
- Chapter 6 presents a lightweight authentication and key agreement protocol for edge computing applications.
- Chapter 7 presents the concluding remarks for the thesis and future work.

Chapter 2

Background and Preliminary

We provide an overview of the edge computing paradigm and its related security issues.

2.1 Edge Computing Paradigm

Since its introduction in 2016 [163], the "edge computing paradigm" term has been interchangeably mixed with other terms and paradigms. Consequently, our goal in this section is to clarify the intermingled terms, review the concept of edge computing, state its advantages, present closely related de-centralized cloud-extension paradigms, and highlight their differences. We also present, herein, several smart applications related to the edge computing paradigm. Fig. 2.1 shows the general framework of different computing paradigms.

2.1.1 Related De-centralized Paradigms

Edge Computing

The OpenEdge consortium defines edge computing as the processing and computations of data at the edge-layer of the network and close to the end-user [198]. Shi *et*

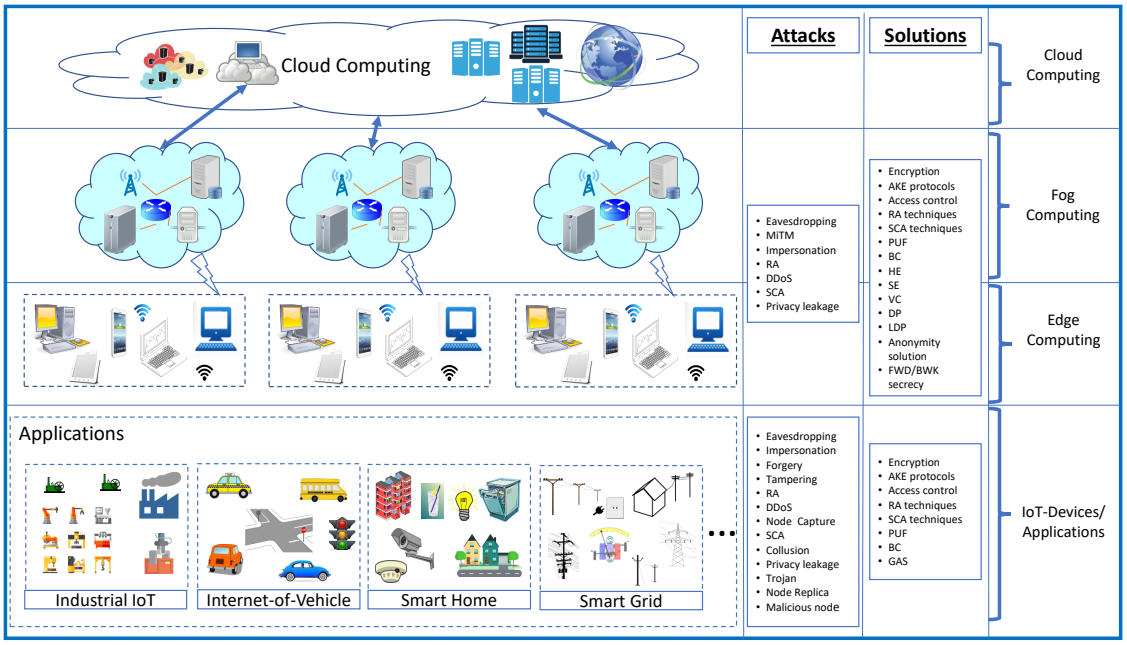

Figure 2.1: Framework of Computing Paradigms

al., on the other hand, define edge computing as any enabling technology that allows computations on the downstream data on-behalf-of the cloud [162]. The motivation of edge computing is that the computation and storage should be closer to the source. It could have been intended initially for edge computing to provide storage and computations only one-hop away from the device generating the data; however, it could be more than one-hop [192]. Nevertheless, the edge layer is a closer point to the end devices than the fog nodes. Edge computing provides more efficiency, less latency, service availability, and control than fog computing due to its location. Edge nodes could be static or mobile; examples of edge computing nodes include, smartphones, small edge servers, WiFi access points, or gateways [88]. In the literature, edge computing may also be referred to as, edge-assisted IoT architecture [8], or edge-centric IoT architecture [158], but in this thesis, we use the term "edge computing."

Fog Computing

Proposed by Cisco in 2012, Bonomi *et al.* argue that fog computing (FC), a virtualized network platform, is advantageous to several IoT applications such as smart cities, smart vehicles, smart grid, and others [36]. They delineate the characteristics of fog computing to include real-time interactions, edge location awareness, scalability, mobility, geographical distribution, heterogeneity, dominant wireless accessibility, and others [36]. Initially fog computing was intended to be a mere extension to Cloud Computing (CC); however, it fast became a paradigm. Moreover, the OpenFog consortium which is the collaboration of industry and academic institutions, later defined fog computing as "A horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum," [135]. The definition implies that fog computing provides all benefits of CC services, functionality, manageability, and virtualization closer to the user in the cloud-to-thing infrastructure. Fog computing can be implemented in several layers of the IoT-fog-cloud network topology; nevertheless, it is a layer between the cloud and the edge layer [88]. Mistakenly, fog computing may be called "edge computing," however the OpenFog consortium in [135] delineates the key differences of both architectures as follows, 1) FC works closer to the cloud while edge computing is isolated from the cloud, 2) FC is a hierarchical paradigm whereas EC works only with few layers close to the end-user IoT-devices, and 3) FC services are not limited only to computations, instead it highly matches the CC services such as networking, manageability, acceleration, and control. On the other hand, Shi *et al.* state that fog computing is focused on the infrastructure side while edge computing is focused on the low-end devices side [162]. Fog nodes could be either static or mobile; examples of fog nodes are small fog servers, access points (AP), controllers, routers, switchers, and others.

Mobile Edge Computing (MEC)

The Internet of Things vision of "anywhere, anytime, and accessibility around the world," concurrently with the advancement of communication devices and ICs, created a myriad of smart applications on smart phones and Internet devices. To efficiently service the massive number of devices connected to the Internet, mobile edge computing was introduced as an extension to cloud computing amalgamated with cellular network technology. Since its creation, MEC has been endorsed by both academia and leading communication and mobile businesses such as Huawei, Intel, Nokia, Vodafone, IBM, and others [5, 74, 88, 139]. In 2014, the European Telecommunications Standards Institute (ETSI) with the Industry Specification Group (ISG) standardized multi-access edge computing (MEC) and defined it as follows: "Mobile Edge Computing provides an IT service environment and cloud-computing capabilities at the edge of the mobile network, within the Radio Access Network (RAN) and in close proximity to mobile subscribers," [74]. The edge nodes in this paradigm could be cellular network (4G/5G) base stations, Internet Service Provider (ISP), or Radio Access Network (RAN). MEC allows for a wide range of applications including, autonomous vehicles, retail, healthcare, smart cities, and others [139].

2.1.2 Edge Computing Advantages

In what follows, we list some of the advantages of the edge computing paradigm as shown in the literature.

Computation

The devices at the IoT application layer are mostly low-end devices with small processing capacities, low storage, and short battery-lifetime. Consequently, these small devices are not capable of storing and processing the data they produce. One of the major advantages of edge computing is computation offloading. Conceptually speaking,

data offloading speeds up the computation process for the low-end IoT-devices. The concept of computation offloading was introduced earlier for other computing paradigms such as mobile-cloud computing (MCC), and the performance has been reported in several publications. However, the early offloading mechanism was concerned with caching the data such as in the Content Delivery Network (CDN) paradigm [162]. On the other hand, in edge computing, both data and operations are offloaded to nearby edge servers. There are several types of offloading, local execution, full offloading, and partial offloading which all are advantageous to the computation process [162].

Communication

One of the most important network performance measures is communication bandwidth. Computation offloading reduces bandwidth consumed by the cloud. Indeed, it is inconceivable for CC to maintain high bandwidth rate with the proliferation of smart applications and IoT-devices.

For example, for a smart camera system in a local neighborhood area, traversing the huge amount of data to the cloud without processing it first, creates a communication bottleneck and consumes large cloud network bandwidth. Thus, having edge nodes close to the vicinity of the rolling-cameras to process and store the data enormously improves performance and saves energy.

Storage

There is a massive amount of data generated at the edge in the three-layer cloud-edge-IoT setting especially with the wide spread of sensors and other low-end devices. With data offloading, edge nodes store the excessive data and process it before traversing it to the cloud. As an example, in [12], the Microsoft research group describes a live Video-analytic project utilizing the edge computing paradigm. The idea of their project is to employ EC nodes to collect, filter, store, process, and analyze live streaming data

received from a significant number of cameras placed around a local area in real-time. The edge nodes in this project are well-equipped and widely distributed in several geographical areas close to the cameras. Consequently, the edge nodes efficiently store and process the massive amount of data, e.g. up to 30 frames/second generated from the cameras [12].

2.1.3 Edge Computing Applications

In this subsection, we discuss smart applications and emphasize their edge computing role.

Cyber-Physical System (CPS)

It is perceived that the cyber-physical system (CPS) and edge computing are two independent paradigms; however, from our study, we see proximity in terms of applications, security, and design. We clarify our findings as follows. First, there tend to be new directions in the scientific research community for incorporating CPS with edge computing to gain design autonomy and security [72]. For example, in a recent publication [205], Zhou *et al.* put forward a framework for energy-trading which integrates CPS with edge computing and blockchain security. On the other hand, Liu *et al.* review research opportunities for autonomous driving and edge computing [115]. Second, edge computing is becoming an integral part of the CPS applications [41, 109, 124, 168]. Indeed, we see large CPS and EC integration especially in the smart grid and transportation applications. Moreover, Zhou *et al.* [205], refer to a new paradigm called, Vehicle to Grid (V2G) which is also an application where CPS and edge computing meet. Finally, we tend to see intersections and overlap between edge computing, IoT, and CPS paradigms [110]. This is due to the fact that CPS is anticipated to be the new generation of embedded-systems that monitor physical objects, and wireless edge computing connects these objects [72]. Thus, we conclude that CPS and EC intersect in the physical world connectivity, applications, design, and security goals.

IoT & IIoT

Similarly, the industrial IoT (IIoT) is very closely related to edge computing and IoT paradigms; IIoT is a subset of the IoT paradigm. Although in the literature, we see the terms, CPS, IIoT, and Industry 4.0, interchangeably used; however, they are different. In [167], Sisinni *et al.* roughly sketch the differences between those three paradigms. For example, consumer-IoT is similar to edge computing, and the end devices are consumer related ones such as smart phones, tablets, and other electronic devices connected to the Internet. Industrial IoT, on the other hand, integrates all aspects of industrial manufacturing with the information technology (IT) that is associated with the business process. Finally, Industry 4.0 main goal is to employ the Internet through smart applications to efficiently expedite the manufacturing process. In the literature, we see edge computing providing optimal solutions for the IIoT/Industry 4.0 in terms of decision-making latency, communication bandwidth, cost reduction, transmission, and privacy improvements [85, 108, 143].

Internet of Autonomous Vehicles

Automatons driving is becoming one of the most attractive research areas in the development of future intelligent transportation systems (ITS). A holistic vision for the smart transportation system is that it lowers traffic deathrate, increases public safety, increases traffic management, and improves public/traffic security. Furthermore, employing edge computing in the Internet of Vehicles framework is necessary for servicing, managing, processing, and storing data produced from the large number of connected vehicles [195]. The framework of this smart transportation is highly heterogeneous. On the one hand, cars are equipped with many embedded devices, sensors, computing nodes, specialized software that connect to the Internet. On the other hand, roadside units, pedestrians, and traffic lights also have their own devices connecting with the Internet. Thus, the infrastructure of all components to operate with each other is highly complicated.

Smart Cities

Urban cities are exploiting all the recent technological and communication advances to raise the quality of living for their citizens. The pervasive Internet devices are now employed to facilitate public services accessibility. There are several challenges to the design of smart city edge computing applications such as, the heterogeneous architecture, the inevitable use of sensors, and the huge data collected from end-devices raise many performance, privacy, and security concerns [120, 197].

2.2 Edge Computing Security

In this section, we classify the edge computing security model into four categorizes, namely, extended EC-security properties, security attacks, countermeasure security primitives, and security protocols analysis methods.

2.2.1 Security Properties

The triad of Confidentiality, Integrity, and Availability (CIA) shown in Fig. 2.2 is a well-known security model defined by Menezes *et al.* [121]. However, in the literature, we see additional security properties for the edge computing paradigm. For example, we find privacy, anonymity, unlinkability, and untraceability which are not included in the CIA security model; nevertheless, these security properties are imperative to the user's privacy in some EC smart applications. Furthermore, there are certain mechanisms that ensure the satisfaction of each property. In what follows, we analyze the EC-security properties and mechanisms.

Integrity

In the three-tier cloud-edge-IoT paradigm, the communication medium among entities in all layers is wireless. Thus, the exchanged messages are easily accessed to all

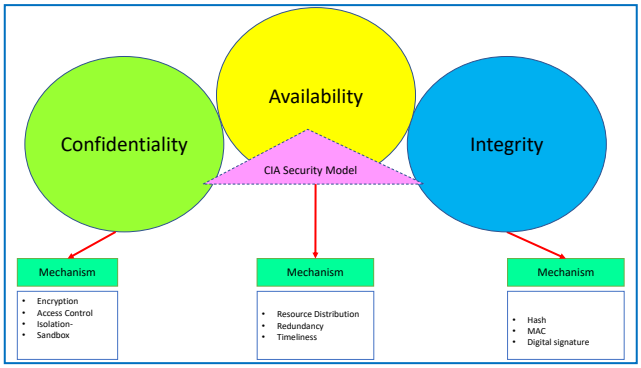

Figure 2.2: CIA Triad Security Model

curious listeners. This exchanged information could be credit-card numbers, social insurance number, or any private confidential information belonging to individuals, governments, or businesses. Consequently, message security is an imperative security property. To guarantee the satisfaction of data integrity designers may utilize, message authentication code (MAC), hash with encryption, HMAC, digital signature, and error correction code [37, 172].

We note here that there are two types of integrity, data origin authentication and data integrity [37]. For simplicity, in this thesis, we use the term authentication for data origin authentication, and for the data integrity we refer to it as message authentication as shown below.

Authentication

Definition 1 *Data Origin Authentication: The cryptographic property of data origin authentication provides assurance that the true identity of the source is as claimed [37].*

In the context of edge computing, authentication could be for mobile-devices, entities, applications, or humans. There are several concerns in edge computing authentication. First, low-end devices have limited resources such as processing power, storage capacity, and battery-lifetime. On the other hand, the edge servers are equipped with good resources. Thus, any EC authentication solution must be lightweight in terms of

both computations and storage requirements to suit the low-end devices regardless of the edge server powers. Second, the standardized communication protocols such as Message Queue Telemetry Transport (MQTT), Transport Layer Security (TLS), and IPsec are resource intensive, and may not be suitable to specific smart applications [172]. Finally, the newly proposed state-of-the-art protocols may have implementations or other security flaws that render them unsuitable for wide usage. In what follows, we list a few authentication mechanisms found in the literature.

i. Mutual-authentication and Key-agreement: The early versions of authenticated-key-exchange (AKE) protocols are not suitable for edge computing because of the low-end IoT-devices. However, to suit the EC framework, we see AKE protocols expand in several directions as follows. First, the initial authentication protocols designed to provide mutual authentication for two parties; however, as the cloud-edge-IoT paradigm exponentially pervaded in the last few years, multiple parties AKE protocols have been proposed to suite authentication for several nodes in the system. Second, we also found recent AKE proposals designed to guarantee other security properties such as forward secrecy and anonymity [2, 19, 173, 182, 187].

ii. Group authentication: In many edge computing smart applications, we see scenarios of large number of IoT-nodes connecting to one edge server. To this end, group authentication schemes was introduced in which a large number of entities gets authenticated all at the same time [20, 51, 71, 105, 128]. Group authentication is suitable for machine-to-machine type of communications and also for massive authentications in MEC and 5G networks [27, 98, 149].

iii. De-centralized delegated authentication: De-centralization is the heart of edge computing. In [190], Yang *et al.* propose a delegated authentication for the vehicle network framework in which the main server delegates edge nodes to collaboratively authenticate vehicles.

iv. Password-based authentication: Other scientific researchers propose password-based authentication suitable for the edge computing paradigm.

Message Authenticity (Data Integrity)

Definition 2 *Data Integrity: The cryptographic property of data integrity provides assurance that the data has not been changed by unauthorized users, entities, or processors [37].*

To guarantee message data integrity, several mechanisms are applied.

i. Message authentication code (MAC): MAC guarantees both the integrity of the data as well as the origin data authenticity [35, 83, 96]. In MAC, both parties share the same key. Another method to ensure data integrity is the hash function with encryption or a Hashed-MAC (HMAC) which is a key-ed hashing [96].

ii. Aggregated-MAC: As an answer to the commonly expected scenario of many nodes requiring authentication from one server, Katz *et al.* in [83] propose Aggregated-MAC. The implementations of aggregated MAC on the edge computing paradigm works as follows. The main server shares a MAC key with each node in the system, and consequently each node uses a regular MAC to authenticate its message. The edge computing server collects the different MAC-tags from each node and aggregates them into one tag using xor operations. Once the main server receives the aggregated message, it can verify all the individual IoT-nodes MAC tags using their shared keys. The aggregated MAC primitive is utilized for group authentication especially in the MEC, LTE, 5G networks [97, 98, 144], respectively.

iii. Message broadcast authentication: With this primitive, each receiver efficiently authenticate broadcast messages without overburdening or communicating with the main server [126, 137, 148, 156].

Confidentiality

Definition 3 *Confidentiality: The cryptographic property of confidentiality provides assurance that the data is accessed only by authorized users, entities, or processors [37].*

Forward and Backward Secrecy

Definition 4 *Forward Secrecy: A protocol is said to provide forward secrecy if the leakage of the long-term key for any participant in the protocol does not compromise the previously established session keys [37].*

In some literature, this definition is extended to include the leakage of long-term and short-term secrets [38]. Furthermore, for group authentication, the definition is reversed by Kim *et al.* in [93] as follows.

Definition 5 *Group Forward & Backward Secrecy:*
Group Forward Secrecy: It is infeasible for an adversary to derive future group keys from a leaked group key.
Group Backward Secrecy: It is infeasible for any adversary to recover previous group keys from the current leaked group keys.

For edge computing applications, forward secrecy is a crucial security property. This is because any leakage of session information should not leak any previous or future session keys. For example, a scenario of an IoT-device theft, should not leak any previous communications. Kaur *et al.* [86] proposed a protocol for mobile edge computing which protects privacy and forward secrecy. Similarly, Jangirala *et al.* [79], utilized the blockchain technology to design authentication and forward secrecy protocol for the RFID edge computing paradigm. Additionally, Zhang *et al.* [199] proposed an authentication protocol with forward secrecy for the edge computing vehicle networks.

Privacy Protecting the privacy of user's information, transferred data, and messages is a crucial part of edge computing smart applications security. Privacy is a major concern to users and private entities especially in the consumer-IoT edge applications. There are several classifications to privacy such as, user's privacy, data privacy, trajectory privacy, and industrial privacy for the IIoT paradigm [59, 81]. On the other hand, there are several developing mechanisms to ensure privacy such as differential privacy [72,81,82,202], location privacy [174], and local differential-privacy obfuscation [189].

Anonymity With the proliferation of edge computing smart applications, anonymity is becoming an important security feature especially for the openly accessed wireless communications. This is because it protects the real identification of the user. Many protocols address mutual authentication and anonymity for the edge computing paradigm. For example, Wang *et al.* [179] proposed an n-times mutual authentication scheme for edge offloading service payments. To protect user's identity in the wide-open wireless medium, they use a fresh pseudo-identity for each communication. Similarly, others use hash-based approaches to protect user's real identity [49, 65, 127, 152, 159].

Untraceability/unlinkability In the same context, unlinkability prevents adversaries or eavesdroppers from linking messages to originators real identities or pseudo-identities. Specifically, in untraceability, 1) any message cannot be linked to the real sender's identity, and 2) any two messages sent in two different times cannot be linked to same origin. We can say that anonymity is a byproduct of unlinkability but the opposite is not true.

For example, Wang *et al.* achieve unlinkability by providing a fresh pseudo-identity in each message communication, and this pseudo-identity is the hash of the original pseudo-identity with the new timestamp [179]. Jia *et al.* [80] use identity-based PKC mutual-authentication scheme that ensures anonymity and untraceability.

2.2.2 Threat Model & Attacks

The edge computing paradigm faces several security threats. We list a few factors that are the root-cause of attacks. First, the wireless communication medium in all edge computing paradigm layers makes all exchanged messages easily accessible to all curious listeners. Second, low-end IoT devices such as sensors and mobile devices are physically accessible to adversaries. Third, there may be several vulnerabilities in the communication protocol itself or the implementation of it. In what follows, we summarize the possible attacks and the security threat model for the edge computing paradigm. We classify the attacks into two types, passive and active.

Passive Attacks, Sniffing, or Eavesdropping

In this type of attack, the adversary just passively listens to all edge communications in the system without interacting with the edge nodes or altering the exchanged messages. However, in weak security protocols, the attacker may infer information or breach user's privacy just by eavesdropping on the communications.

Active Attacks

Man-in-The-Middle (MiTM) Attack In this type of attack, the adversary stands in the middle between two honestly communicating edge nodes and maliciously modifies messages. Each honest edge node receives messages from the adversary thinking they are from the other honest node. Protocols that are based on PKC are more susceptible towards this type of attack than symmetric-based security protocols.

Impersonation Attack There are two types of impersonations, the adversary may impersonate a valid edge node in the system to pass the edge-server authentication test, or the adversary may impersonate an edge-server to pass the node's authentication test.

Tampering The attacker may take advantage of the wireless communication medium and tamper with message delivery by either delaying or dropping the messages to degrade the quality of service of the network.

Replay Attack This is also called a freshness attack in which a malicious adversary records messages for a period of time and replays these messages at a later time. Communication protocols that lack freshness guarantee are vulnerable to this attack.

Distrusted Denial-of-Service (DDoS) attack There are several types of DDoS attacks, but the traditional one works as follows. A malicious attacker sends streams of legitimate dispensable packet-requests that jam the target edge server or flood the edge node to exhaust all its hardware resources such as battery and storage space. Some types of DDoS include, battery draining attack which attempts to drain the power/battery resources of the edge node. On the other hand, in the DDoS-outage attack, the adversary hinders the affected nodes from operating regularly.

Node Take-over The low-end devices in the edge computing paradigm like sensors, wearable devices, and mobile devices are physically accessible to the attacker. Thus, the attacker may steal these devices to obtain network information such as shared keys, tamper with the integrated circuit (IC), modify the operating system, change the running software, or access cryptography information [125]. These attacks usually corrupt the whole device but gain information to attack the network.

Side-Channel-Attack (SCA) In this attack, the adversary depends on the correlation between the publicly accessible information and the private information. The differential analysis of encrypted data consequently deduces parts or all of the user's private secrets. Kocher *et al.* in [95] use differential power analysis to recover private secrets from protected devices. One of the SCA countermeasure solutions is solving multivariate equations

signature scheme [130].

Collusion Secret sharing schemes are vulnerable to collusion in which several edge nodes collude with each other to retrieve the shared secret.

Hardware-Trojan This is a physical attack on edge devices in which the attacker physically accesses the IC of the edge node to retrieve its data or run malicious software.

Privacy-leakage This represents the leakage of data that carries private information of the user.

Node-replication This attack is done by injecting a malicious node in the system with the same ID as an existing node. This enables the adversary to redirect packages, steal, and corrupt information received to the original edge node.

Corrupted/malicious-nodes Corrupted/malicious edge nodes are those that can obtain un-authorized access to the edge server or other edge nodes in the network [125]. Consequently, the attacker can direct these to control the network, inject malicious messages, or block messages.

2.2.3 Cryptography Primitives

Un-keyed Primitives

Un-keyed primitives play a vital role in edge computing security solutions. This is because un-keyd functions are considered lightweight and require no shared keys between parties. Indeed, a pre-shared key maybe liable to leakage especially in the low-end devices such as sensors and mobile devices. For example, in a node-take-over attack, the adversary may access all the stored data in the device's memory including long-term keys and

session keys. Examples of unkeyed primitives are the one-way functions such as the Hash functions; SHA-1, SHA-256, and SHA-128 [64].

Keyed Primitives

There are two types of encryption algorithms, symmetric-key based and public-key based.

Symmetric-key Based Primitives In symmetric-key cryptography, both communicating parties share the same key, and this key is used for both encryption and decryption. For the edge computing paradigm, symmetric-key based protocols are lighter than public-key based protocols in terms of computations and communication overhead. This is because they depend on simpler encryption algorithms such the Advanced Encryption Standards (AES) and others.

Public-key Based Primitives Public-key based algorithms such as RSA, DH Key exchange (DHKE), and El-Gammal encryption, require large number of computations and storage capacity which render them unsuitable for low-end devices. Nevertheless, Elliptic Curve Cryptography (ECC) [70] is considered a lightweight in comparison to the other algorithms. Hence, for edge-computing, several lightweight schemes based on hash function in combination with ECC are proposed. For example, Wazid *et al.* [181] propose an xor-based ECC lightweight scheme for Internet of Vehicle paradigm; while Amor *et al.* in [11] propose an ECC based mutual authentication scheme for the edge/fog computing environment. Furthermore, Khan *et al.* in [88] conduct a survey comparing lightweight protocols suitable for the IoT-paradigm.

Secret Sharing Schemes

The fundamental idea of secret-sharing schemes introduced independently by both Shamir and Blakely in [30, 160], respectively in the same year, 1997. It is basically based

on the security of polynomial interpolation over a finite field. It is called (t, n) threshold Shamir secret-sharing scheme, and the mechanism works as follows. A trusted dealer selects a (t-1)-degree polynomial, and the secret is the intersection of this polynomial with the x-axis. The polynomial is kept secret, and the dealer generates the secret-shares which are points on the polynomial and distributes them through a secure channel to each group member. To reconstruct the secret, at least t shareholders must reveal their shares and apply Lagrange's interpolation formula.

Access Control

One of the fundamental mechanisms for attack prevention and assurance of the confidentiality property is access control. It is the process of restricting and managing resource/access to only privileged users or entities. In the setting of the three-tier cloud-edge-IoT computing paradigm, access control faces several challenges. First, accessing the storage and computational cloud resources require policies and privileges. Second, the edge or fog nodes need to have the same privileges and policies as the cloud. Third, virtual machines (VM) also require access control policies to avoid attacks, [200]. There are several access control models proposed in the literature. Aleisa *et al.* [6], review and discuss the different access control models, aspects, and challenges for the fog computing paradigm. They present the different access control models such as attribute-based-access-control, discretionary-access-control, role-based access-control, access-policy-access-control, identity-based-access-control, task-based-access-control, rule-based-access-control, mandatory-access-control, and state-of-the-art access control model for fog-computing.

2.2.4 Security Protocols Analysis Methods

There are several types of formal security analysis used for the EC-paradigm. However, in what follows, we provide an overview of the ones used in our thesis to formally prove proposed protocols.

Theoretical Methods

CK Model: The Canetti-Krawczyk (CK) model is a comprehensive security threat model which aims at formally analyzing communication security protocols to mimic real-world attacks [38]. The attacker in the CK-model is a probabilistic-polynomial-time (PPT) adversary which has full access to all communications between system's entities. Specifically, the adversary can listen, inject, drop, re-direct, and delay message delivery for all communications in the system. Additionally, the CK-model allows the adversary to obtain state session, session key, and internal memory through successful attacks. In *session-state-reveal*, the adversary is able to access all the current state session information except those which are directly related to accessing the long-term key. However, the *session-key-query* enables the attacker to reveal old session keys. This resembles a real-world attack of cryptanalysis or insecure old session key removal. Finally, the *party-corruption-attack*, allows the adversary to access all the entity's internal memory. This resembles a real-world node theft.

On the other hand, the proof of semantic security of any cryptography primitive is usually defined as an indistinguishability-game, also called an attack-game, played between two entities, namely, a malicious adversary and a decent challenger. The security proof of the entire communication protocol is usually organized as a sequence-of-games (SoG), [166]. Together, the CK-threat model and the SoG create a proof mechanism approach that is widely adopted for proving protocol security. In [133], Odelu *et al.* formally prove the security of their smart grid authenticated-key-agreement protocol using both the CK-model with the SoG attack organization. Similarly, Lee in [102], formally proves his anonymous authentication protocol for the distributed computer networks with CK-model and SoG.

RoR Model Similarly, the Real-or-Random (RoR) model [4] is a formal security model. It is similar to the CK-model. However, it is designed to prove the semantic security of

exchanged messages in the three-party password-based authentication and key-exchange (AKE) protocols. The following edge computing schemes combine RoR with SoG to prove the semantic security of the protocols [46, 69, 152, 181, 183].

Automated Methods/Tools

Additional to the above methods, protocol designers are using formal automated tools for verification of their designs. The most used ones are the Automated Validation-of-Internet Security Protocol and Applications (AVISPA) [18], Scyther [154], ProVerif [141], Tamarin [171], VerifPal [94], and Alloy [7].

In what follows, we briefly present the automated tools used in this thesis.

AVISPA is an automated software widely used for security protocol verification [14, 18, 170]. The tool searches for potential attacks such as Man-in-The-Middle attack, replay attacks, impersonation, active and passive attacks according to the Dolev-Yao (DY) intruder model [57]. For more details, the interested reader may refer to the following references [15, 18, 26, 170, 175, 177].

The flow of analyzing the security of a protocol in AVISPA is as follows.

A) The security protocol is first described in the High Level Protocol Specification Language (HLPSL) [177] along with the specified security goals. HLPSL is a role based description language in which each entity in the protocol is represented by a sequence of defined states. The protocol session is described in two top-level roles called the "session role" and "session role," and the security goals are defined in a special role.

B) Using the HLPSL2IF translator, AVISPA converts the HLSPL protocol code into Intermediate Format (IF) to feed it into one of the four back-end protocol analyzers integrated in the AVISPA software, namely, (i) On-the-Fly Model Checker (OFMC), (ii) the Constraint-Logic-based ATtack SEarcher (CL-AtSe), (iii) SAT-based Model

Checker (SATMC), and (iv) Tree-automated based on automatic Approximation for Security Protocols (TA4SP). The back-end analyzers share some basic security model, specifically the Dolev-Yao (DY) intruder model [57], however, they have different attack searching mechanism. The details and functionality of these back-end analyzers are available in [15, 18, 26, 175].

C) The output format (OF) of the back-end simulation shows the result of the protocol security analysis, specifically, SAFE, UNSAFE, or inconclusive for the set of the specified security goals in the HLPSL protocol description. If the result is UNSAFE, AVISPA provides an attack scenario. AVISPA comes with an animated software tool, namely, Security Protocol ANimnator (SPAN)+AVISPA [170].

Verifpal is a formal verification tool [94] which is considered a spin-off ProVerif [31, 32, 32]. Verifpal is a symbolic-model tool in which the adversary is a probabilistic-polynomial-time 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 [25]. ProVerif semantics are also based on Coq [31, 32, 32], while AVISPA's semantics are based on Lamport's Temporal-Logic-of-Actions [101].

We note here that most of the automated tools have limited build-in cryptography primitives. For example, to the best of our knowledge, the SSS primitive is only available in VerifPal. Also, most of the automated tools do not support algebraic operations such as addition and multiplication. Therefore, analyzing protocols with advanced and hybrid security primitives using automated tools may not be possible. For these protocols, security analysis would be performed using some of the aforementioned models such as the

RoR-model.

2.3 Utilized Security Primitives

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

2.3.1 Hash Function

Definition 6 *A hash function must satisfy the following properties [151]:*

- *Preimage-resistance: For the hash function, $y = H(x)$, it is infeasible to find any input $x^{'}$ that hashes to the same y; i.e. knowing y, it is infeasible for an attacker to find a preimage $x^{'}$ such that $H(x^{'}) = y$.*
- *2^{nd}-preimage-resistance: It is infeasible to find any 2^{nd} input, $x^{'}$, such that $H(x) = H(x^{'})$ where $x \neq x^{'}$.*
- *Collision resistance: It is infeasible to find two different inputs x and $x^{'}$ that hash to the same output.*

2.3.2 Secret Sharing Schemes

In the (t, n) Shamir's secret-sharing scheme [160], a trusted dealer splits a secret s into n shares, $n \geq t$, where n is the total number of shareholders and t is the threshold. The mathematical principle of binding n shares is a (t-1)-degree polynomial $P(x) = s + a_1x^1 + \cdots + a_{t\text{-}1}x^{t\text{-}1}$, such that $P(0) = s$ is the secret, and each shareholder obtains its (x_i, y_i)-point secret-share on this polynomial via a secure channel. To recover the secret, at least t out of n shareholders reveal their tokens and compute the Lagrange's interpolating polynomial.

Definition 7 *A secret-sharing scheme over $\mathbb{Z}_q$ is composed of two algorithms, namely $\mathcal{G}$ and $\mathcal{C}$.*

- *$\mathcal{G}$ is a probabilistic algorithm that generates t-out-of-n shares of k. $\mathcal{G}$ is invoked as $\mathcal{G}(n,t,k) \xrightarrow{R} (s_1, s_2, \cdots, s_n)$ 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.*
- *$\mathcal{C}$ is a deterministic algorithm $k \leftarrow \mathcal{C}(s_1^{'}, s_2^{'}, \cdots, s_t^{'})$. It is invoked to recover k using the Lagrange's interpolation formula.*
- *Correctness: For every t set of shares of k, $\mathcal{C}(s_1^{'}, s_2^{'}, \cdots, s_t^{'}) = k$.*

On the other hand, in the (n, t, β) multi-secret sharing schemes, a group of at least t out of n participants share their secret-shadows to recover β secrets. Most of these schemes [53, 73, 136, 161, 191, 201] are based on Shamir 's secret sharing scheme [160] and a two-variable one-way function $f(r, s)$ [73]. The properties of the two-variable one-way function, as shown in [191], 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, 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 second protocol, we utilize Yang's *et al.* scheme [191] with $k = 1$ and the two-variable one-way function to implement a group authentication scheme at the edge of the network.

Secret Sharing Homomorphism

The homomorphism property for the SSS is defined as follows [28].

Definition 8 *Let S be the set of all possible-secrets, and let T be the set of all legitimate-shares, respectively. Let $F_I \colon T \rightarrow S$ be an induced function, such that $s = F_I(s_{i_1}, s_{i_2}, s_{i_3}, \cdots, s_{i_t})$ where I is any subset of T. Let $\oplus$ and $\otimes$ be two-binary functions conducted on the sets of S and T, respectively. A (t, n) threshold scheme is $(\oplus, \otimes)$-homomorphic if $(s \oplus s^{'}) = F_I((s_{i_1} \otimes s^{'}_{i_1}), (s_{i_2} \otimes s^{'}_{i_2}), (s_{i_3} \otimes s^{'}_{i_3}), \cdots, (s_{i_t} \otimes s^{'}_{i_t}))$.*

The (t, n) Shamir's secret-sharing scheme is $(+, +)$-homomorphic. In our protocol, we take advantage of this property which states, informally, the addition of the two SSS-polynomial secrets, equals the Lagrange's interpolation polynomial of the addition-of-shares for the same subset I.

2.3.3 Aggregated Message Authentication Codes (*Aggregated-MAC*)

In [83], Katz *et al.* propose an aggregated message authentication code, *Aggregated-MAC*, and prove that its security properties are analogous to the security properties of the standard MAC.

Definition 9 *Aggregated MAC is a set of probabilistic-polynomial-time (PPT) algorithms, namely, $(MAC, Aggregated\text{-}MAC, Verify)$, defined as follows.*

- *MAC: Let $\kappa \in \{0,1\}^{\lambda}$ be a key with a length equals to the security parameter, λ, and let $msg \in \{0,1\}^{*}$ be any arbitrary length message, MAC is constructed from the standard keyed pseudo-random function, $tag \leftarrow MAC_{\kappa}(msg) = \mathcal{F}_{\kappa}(msg)$ where $\mathcal{F}$ is a pseudo-random function.*

- *Aggregated-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. $M = \{(msg_1, id_1), (msg_2, id_2), (msg_3, id_3), \cdots, (msg_l, id_l)\}$, and the corresponding tag is constructed by simply XOR-ing all the messages tags,*

$tag = tag_1 \oplus tag_2 \oplus tag_3 \oplus \cdots \oplus tag_l$.

- *Verify: Given the set of all identifiers keys* $\{\kappa_1, \kappa_2, \kappa_3, \cdots, \kappa_l\}$*, and the message tag pair,* (M, tag)*, the verify algorithm,* $Verify_{(\kappa_1,id_1),(\kappa_2,id_2),\cdots,(\kappa_l,id_l)}(M, tag)$ *outputs* 1 *upon successful authentication; and* 0 *otherwise.*

2.3.4 Physically Unclonable Functions

Introduced in 2002 by Gassend *et al.* [63], Physically Unclonable Functions aims at creating a digital fingerprint to authenticate Integrated Circuits (IC) devices. Silicon PUF relies on the fact that there are different natural variations in the IC manufacturing process which makes each IC chip unique such that no two different IC devices can identically have the same physical structure. In edge computing, PUF is used for authentication, and the typical PUF-based authentication protocol relies on collecting a large set of challenge-response pairs from different PUF-implanted nodes and stores them in the authenticating server's database. Specifically, when the authenticating server requires a node authentication, it selects a challenge from its database and sends this challenge to the IoT node. If the challenge-response pair matches the stored value in the server's database, the node is authenticated; otherwise, it is rejected. Because of the unique PUF response, the probability to impersonate a node by predicting its PUF-response is highly unlikely; thus, authentication is achieved. In a subsequent time, another challenge-response pair is selected for authentication. In the literature, there are several proposals for lightweight authentication protocols based on PUF, designed specifically for low-end devices which are suitable for edge-IoT-devices.

Definition 10 *Let* $C \in \{0,1\}^\lambda$ *be a string of bits of length* λ*, and let* PUF *be a deterministic function such that* $R = PUF(C)$*. We say that* PUF *is a secure physically unclonable function if the following holds.*

- A response to a challenge $R_i = PUF(C_i)$ *gives negligible information to another re-*

sponse $R_j = PUF(C_j)$ *where* $i \neq j$.

- *Without physically having the PUF-device, it is infeasible to generate* $R_i^{'} = R_i$ *where* $R_i = PUF(C_i)$.

- *If an adversary,* $\mathcal{A}$*, tampers with the PUF-device, the PUF function is destroyed.*

Noisy PUFs and Fuzzy Extractors

Using physical unclonable functions as the underlying security primitive in a protocol faces many challenges [61]. This is mainly because, ideally, anytime a PUF is queried with a challenge, it is expected to produce the same response. However, in reality, PUFs are susceptible to noise such as voltage fluctuation, temperature, and other operating/environmental conditions. Specifically, exciting a PUF with a challenge C may not always return the same raw response Raw, the output could be a $Raw^{'}$ in which the error is defined as $e = Raw \oplus Raw^{'}$. Consequently, using a PUF-response as a cryptography-key may jeopardize the operation of the security protocol. To circumvent the PUFs inherent noise problem, fuzzy extractors (FE) and (reverse) fuzzy extractors (rFE) are introduced in the literature [56, 62, 116, 176].These solutions reconcile the bit-errors and stabilize the PUF-responses. Additionally, some of these solutions are suitable for low-end devices such as RFID-tags and wireless-sensors.

Fuzzy Extractors: Using Fuzzy-Extractors (FE) stabilizes noisy PUF-responses to be reliably used as cryptographic keys. The FE model is composed of two algorithms. Algorithm 1 is a key generation function and Algorithm 2 is a reconstruction function, $FE.Gen(.)$ and $FE.Rec(.)$, respectively [56, 178]. The $FE.Gen(.)$ is a probabilistic function that takes the raw PUF-response Raw and generates a key and a helper-data vector. The key in our protocol is the node's secret-share in the Shamir's SS scheme $\mathbf{R}$ and the helper-data $\mathbf{hd}$ is a vector determined from the original PUF-response Raw and a parity-check matrix of a linear-error-correction-code $\mathbf{ECC}$, [62, 178].

Thus, $< \mathbf{R}, \mathbf{hd} >= FE.Gen(Raw)$.

Algorithm 1: The Gen Algorithm $FE.Gen(.)$

Input: A challenge C.
$Raw = PUF(C)$
$\mathbf{R} \xleftarrow{\$} \mathbb{Z}_n$
$\mathbf{hd} = Raw \oplus \mathbf{ECC}(\mathbf{R})$
return $< \mathbf{R}, \mathbf{hd} >$

On the other hand, the $FE.Rec(.)$ is a deterministic function that uses the helper-data **hd** with the noisy $Raw^{'}$ that is close to the original Raw to reproduce the key. The reconstruction function applies error-decoding algorithm **D** to recover $\mathbf{R} = \mathbf{D}(Raw^{'} \oplus \mathbf{hd})$.

Algorithm 2: The Rec Algorithm $FE.Rec(.)$

Input: C; helper-data string **hd**.
$Raw^{'} = PUF(C)$
$\mathbf{R} = \mathbf{D}(Raw^{'} \oplus \mathbf{hd})$
return $< \mathbf{R} >$

PUF Types

PUFs are typically classified in the literature in terms of their hardware fabrication process and security strength. In terms of their construction methods, there are two major types, silicon-based and non-silicon based PUFs. Taking advantage of the Complementary-Metal-Oxide-Semiconductor (CMOS) discrepancies, the silicon-based PUFs are constructed with the aid of different circuit architectures. As an example, the Ring-Oscillator based PUFs, Arbiter (time-delay) based PUFs, SRAM based PUFs, Butterfly (Bi-stable circuit) based PUFs, and Delay PUFs are types of silicon-based PUFs and optical PUFs are examples of non-silicon based PUFs [61, 118].

On the other hand, in terms of security levels, PUFs are classified into strong PUFs, weak PUFs, and controlled PUFs. The threat model for this classification is based on the adversary's ability to access the PUF's challenge and response space.

1. **Weak PUFs:** They have very limited CRP space, and sometimes only one. The response space may never be revealed. Weak PUFs are typically used for deriving a protocol secret key. Examples of this types are SRAM-based PUFs, Ring-Oscillator-based PUFs, Arbiter-based PUFs, and Buterfuly-based PUFs [178].

2. **Strong PUFs:** The CRP space is very large and impossible to collect in a reasonable period of time. Thus, they are more robust and resist more attacks. They are typically used for different cryptography applications such as authentications. The bulky non-silicon-optical PUFs and the super-high-information-content SHIC-PUFs are considered strong PUFs that have not reported any modeling attacks vulnerabilities [61].

3. **Controlled PUFs:** Controlled PUFs are strong PUFs with additional control-logic circuitry proceeding the core-PUF interface to implement more functionalities.

Chapter 3

Lightweight Broadcast Protocol

3.1 Introduction

The need for improving city services is becoming a necessity and not a luxury as previously perceived. For example, enhancing city emergency services saves lives and mitigates physical damage. Numerous disastrous situations could have been prevented with fast response to emergency calls. To provide better services, researchers are utilizing the new rapidly booming technologies, IoT, FC, EC, and Fog/Edge Computing (FEC) paradigm in smart applications. FEC is gaining a great momentum among researchers for meeting performance requirements and eliminating communication delays for selected applications [36, 134, 135, 162, 194]. This paradigm addresses the performance of real-life scenarios of things connected to each other and to the Internet.

On the other hand, providing security for the FEC paradigm remains a challenging problem. Fog/Edge paradigm entails connecting large number of things and ubiquitous sensing components to a few fog/edge nodes. Furthermore, some of the entities in the paradigm are low-end devices with small storage capacity, have low energy sources and under-powered processors. Moreover, with the recent advances in physics that point towards the eventual construction of large-scale quantum computers [16], most of the

existing solutions will become insecure [42, 60, 164]. Precisely, broadcast protocols whose security relies on the hardness of factoring and finding discrete logarithms in finite fields are broken in polynomial time using Shor's algorithm on quantum computers [165]. Thus, applying conventional security solutions may not be suitable for these new smart city paradigms. To this end, we propose a hash-based lightweight secure *FEC-IoT* broadcast message protocol whose underlying cryptographic blocks have security properties that are easily comprehensible, plausible, and are proven to be secure against quantum adversaries [169]. Our case study in this chapter is a smart emergency application that can save lives and prompt immediate smart emergency responses when needed.

3.2 Related Work

The main focus of our work is to provide authentication and security for communications between fog/edge devices with the end users, i.e. things. Our platform in this proposed protocol is shown in Figure 3.1; the Fog/Edge architecture shows that the network is divided into three layers: cloud servers, fog and edge devices, and things. Things can be small sensor devices connected to different entities depending on the application.

Indeed, edge/fog computing is a recent network paradigm which has many security concerns. Addressing these issues using classical design methodologies is not sufficient and does not meet the required real-time constraints. For example, designing protocols which achieve authenticity is a challenging problem. Public key-based solutions suffer from heavy computations and large bandwidth requirements [42, 140, 147, 164]. Solutions which utilize time for asymmetry, such as μTESLA and its variants [111–113, 137], also suffer from delays and vulnerability against DoS attacks. While one-Time-Signature (OTS) schemes [44, 106, 123, 131, 138, 150, 180] are attractive in terms of computational requirements, they suffer from scalability and large signature and public key sizes. Even though solutions based solely on one symmetric key are very energy efficient, they have their obvious

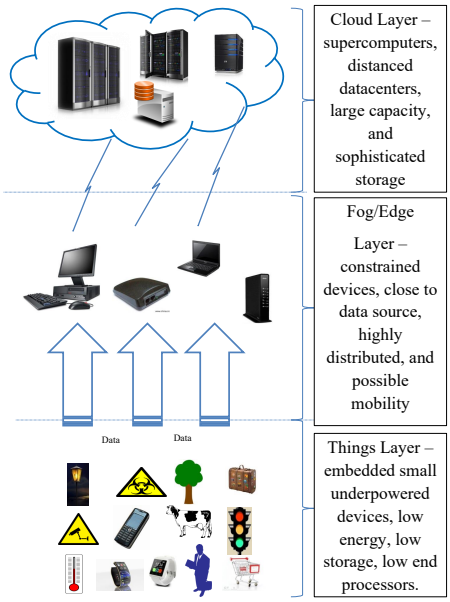

Figure 3.1: Fog/Edge Computing Architecture

security disadvantage: the compromise of a single node can compromise the security of the entire network. Our solution mitigates this problem and ensures that a compromised node affects only the confidentiality of a local cluster and not the entire network.

To the best of our knowledge, a solution for message authentication in the context of the considered Fog/Edge broadcast scenario has not been considered before. However, in here, we list some proposals in the authentication of edge computing in general. Kim *et al.* [90–92] proposed a distributed scheme which uses local authorization entities based on globally distributed trust among these entities. The locally centralized entities keep the credentials of registered devices locally stored in its database. These entities manage the authorization locally, which is achieved by distributing session keys and setting specific access activities. In [76], the author used a symmetric-key based protocol for a dynamic Fog/Edge paradigm. The scheme requires users to hold one long lived master key, and the fog servers are required to store one secret key for each user. The approach is primitive and requires a large communication overhead, especially for group broadcasting or mul-

ticasting of a single message. Recently, Wang *et al.* [179] proposed a lightweight mutual authentication scheme for edge offloading. However, the scheme is for smart card holders and requires pre-shared keys and passwords. Similarly, Wazid *et al.* [181] proposed a key management scheme for Internet of Vehicles (IoV) with very light computations, but also requires pre-shared keys and passwords. Also, several key agreement protocols have been recently proposed in the context of vehicle to grid (V2G) networks (e.g., see [3]). For the purpose of comparison, we select from literature lightweight protocols designed for low-end devices, specifically [1, 42, 60, 104, 106, 114, 140, 164, 179, 181, 206]. A brief summary of each protocol, its advantages/disadvantages is summarized in Table 3.1.

In [60], Fouda *et al.* describe a lightweight hash-based message authentication protocol for smart grid networks. Though their algorithm is a combination of PKC and symmetric key based solution, we list it here for its related features. Their solution is solely designed for unicast communication where smart grid meters inside Home Area Networks (HAN) communicate with the station in Building Area Networks (BAN) using hash-based authentication scheme. Their hardware platform is a smart meter with a 16 MHz processor and 8KB RAM capacity and the authentication protocol is based on establishing a session key using DH key exchange. The computational complexity of their approach is in establishing the session key through DH key-exchange protocol. The messages, on the other hand, are delivered with regular AES encryption and authenticated with MAC. The number of nodes in HAN is between 20-140 nodes. Similarly, Li *et al.* [104] propose a smart grid secure HAN Merkel-Tree-Based authentication scheme where the smart meters have the same specifications as described in [60]. Liu *et al.* propose a secure key management scheme for Advanced Metering Infrastructure (AMI) which supports all types of communications, unicast, mutlicast, and broadcast. They use a method similar to ours where they group the nodes according to the Demand Request. In [106], the protocol is a one-time-signature (OTS) based on HORS signature [150]. A recent efficient advanced smart metering secure communication protocol is proposed in [1] where the

mutual authentication is based only on hash function and random number generation; however, the protocol has large communication overhead. Leap [206], on the other hand, is a localized encryption authentication protocol with multi-keying symmetric key which supports all communication types. The protocol involves four keys: a pair-wise key with neighboring nodes, cluster key, group key, and individual keys with the base station. A group key is used for broadcasting messages and pair-wise keys are used for a node to communicate with its neighbors. The processor used is Mica2 Mote sensor processor [29], with 8MHz CPU and 4KB RAM. One of the disadvantages of their scheme is the amount of storage that each node has to hold. For example, if a node has 100 neighbors, then the number of keys will be $100 \times Number\ of\ bytes$ for pair-wise keys. Porambage *et al.* [140] present a Two-Phase authentication scheme based on Elliptic Curve Cryptography (ECC) and a third-party certificate for authentication. IMBAS and εIBAS [42,164] are two ECC-based PKC message broadcast authentication schemes using pairing-free ID-based, and pairing-optimal ID-based BA, respectively. The sensors used for their design are also based on Mica2 Mote and the number of nodes can scale up to thousands. Similarly, our broadcast scheme is also independent of the number of nodes. Thus, the number of nodes can scale up to a large number.

3.3 Protocol Applications

Our protocol is suitable for networks or systems that require lightweight computation, low energy, small storage, and high level of security. Particularly, the protocol is based on a one-way hash function for session key generations which requires less computations than other protocols. Also, the symmetric key nature requires less computations than the public key cryptography. Light computations are especially important for small low-end devices that are limited in power such as portable devices and sensors. Furthermore, most of the low-end devices have small under-powered processors with small

Table 3.1: *FEC-IoT* Related Protocols Summary

Reference	Application & Utilized Schemes	Security Goals and Advantages	Disadvantages
This work	– edge computing – broadcast scheme – symmetric key cryptography	– authentication, confidentiality, integrity, forward secrecy – light computation and storage	– pre-shared key is required
LAMANCO [179]	– edge computing offloading – unicast mutual authentication – Symmetric+ ID-Based + bilinear map	– authentication, confidentiality, integrity, user anonymity – light computation and storage	– pre-shared key is required.
LWMA [60]	– smart grid network – unicast scheme – DH+hash-based function	– authentication, confidentiality, integrity, forward secrecy – small storage	– large computation in DH key exchange
ULSS [1]	– smart grid network – bi-directional comm. scheme – Hash and XOR functions	– authentication, confidentiality, integrity – very light computations – small storage	– large communication overhead
M-Tree [104]	– smart grid network – unicast scheme – Merkel-Tree based authentication	– authentication, confidentiality, integrity – light computation	– large storage – large communication overhead – no forward secrecy
KMS-AMI [114]	– AMI – all types communication scheme – Hash and XOR functions	– authentication, confidentiality, integrity, forward secrecy – light computation	– large number of keys – large communication overhead – large cost in key distribution – broadcast-key leak comprises network – security properties not formally proven
LEAP [206]	– Localized Encryption and Authentication Protocol – supports all types comm.	– authentication, integrity – light computation – low communication overhead	– large number of pair-wise keys – no forward secrecy
TSV [106]	– Tunable signing and verification – One-Time-Signature scheme	– authentication, integrity – light computation – fast verification	– large signature – large public key size – low security level, 80 bits – not scalable
IMBAS [42]	– authentication protocol for WSN – broadcast communication – ID-Based multi-user	– authentication, integrity – scalable – secure Schnorr signature	– computationally intensive – no confidentiality – large communication overhead – low security level, 112 bits.
εIBAS [164]	– authentication protocol for WSN – broadcast communication – ID-Based pairing-optimal	– authentication, integrity – scalable – reduced communication overhead	– computationally intensive – no confidentiality – low security level, 112 bits.
2-Phase [140]	– authentication protocol for WSN – unicast communication – 2-phase authen. certificate-based – based on secure ECC	– authentication, confidentiality integrity	– third-party certificate – computationally intensive – large communication overhead
AKM-IoV [181]	– Internet of Vehicle (IoV) network – authen. key management scheme – ECC + ID-Based + XOR	– authentication, confidentiality, integrity, user anonymity – very light computations – authentication using Hash only	– large computation in initialization phase

memory which makes them incapable of executing complex algorithms. IoT and Industrial IoT (IIoT) applications can also benefit from our lightweight protocol in scenarios where the IoT device has limited processing capabilities. Thus, even if edge-nodes have advanced capabilities, the low-end IoT nodes need a lightweight algorithm for extended battery and energy lifetime. Examples of networks that can benefit from our lightweight authenticated broadcast protocol are smart grid networks where the smart meters in the HAN and GW are usually composed of micro-controllers such as MSP430F471xx with $\sim$ 16MHz CPU, 8KB RAM, and 120KB flash memory. Although the GWs in the Neighborhood Area Network (NAN) GW have more processing capabilities, their counterpart smart meters have limited computational resources. Therefore, our protocol can be used in the broadcast between NAN GW and HAN GW in smart grid networks. Our protocol can also be used in smart transportation fleet management systems who's expected effi-

ciency, in terms of response time, can benefit from the decentralized fog/edge computing paradigm. Edge entities can download the traffic management data from cloud servers, carry out location-based processing, then broadcast the processed data to the intended fleet member. We henceforth present a smart medical emergency application as a case study to illustrate our protocol.

Case Study: Smart Medical Emergency Application

The considered FEC application scenario is depicted in Figure 3.2, where we provide a secure authenticated broadcast message to the different groups shown in the Figure. In particular, we consider an edge device connected to several areas, i.e. neighborhoods, in a city as well as different officials or related personnel such as police officers or family doctor. Several of these edge-emergency systems can be spread around in different neighborhoods where all edge entities connect to the main cloud server. To gain performance, the edge entities act immediately to an emergency incident, however, all emergencies and incidents are later reported to the main cloud. In this model, sensor nodes mounted on streetlights, intersections, public parks, vehicle parking lots, nursing homes, buildings, and residential housing area, continuously stream recorded data to the edge entity. In addition, sensor tracking devices can be distributed to citizens with special needs, for example, patients, elderly people, and those who have permanent or temporary disabilities. The edge entity will process these data and if needed will send information securely to one of the service groups such as medical groups, firefighter department groups, law enforcement groups, volunteer or social worker groups. There could be other groups; however, as an illustrated example, we list only four in Figure 3.2. For example, if a patient or a disabled person accidentally falls on the street or at her home, the edge entity will send the patient's information to the medical group. Subsequently, the medical group which consists of nearby hospitals, family doctors, clinics and ambulances will be notified. The medical group members will then search for the patient's records and send the appropriate team

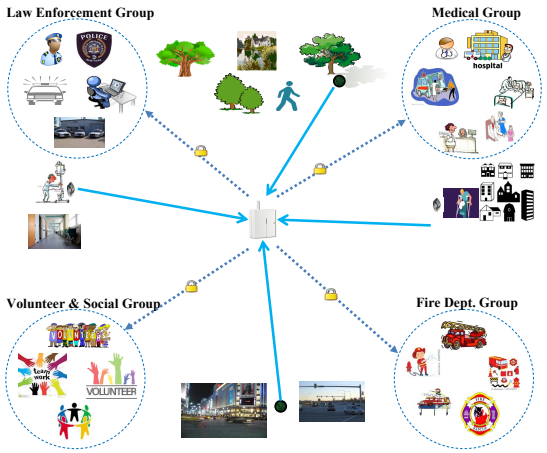

Figure 3.2: Smart Medical Emergency FEC Application

to the patient's location. Another scenario is at a street intersection. If an accident occurs, the edge will notify different groups to act to the scene of the accident. By notifying different groups, the closest law enforcement unit or ambulance can respond to this accident to prevent life losses.

Figure 3.2 shows several nodes connected to the system where these nodes are classified as streaming nodes and receiver nodes. Streaming nodes are those connected to the edge entity and constantly streaming recorded data to the edge. For example, CCTVs, security cameras mounted on street intersections, monitoring devices held by patients and those with special needs are considered streaming nodes. The other type of nodes are receiver nodes, and they are, for example, entities representing officials at different institutes such as hospitals, clinics, or police stations. In our application, there could be a scenario where one incident requires communication with several different groups.

The edge entity groups the receiver nodes based on the preferential interest of the end user. For example, all nearby clinics, hospitals, or volunteer healthcare providers are grouped into one group for a patient with a tracking device. Similarly, all nearby registered police stations or police vehicles are grouped into one group for a street intersection. Table

Table 3.2: Examples of Grouping

End user	**Associated Groups**
Elderly person	Medical group, Family group
Street intersection	Law enforcement, Firefighter groups
	Medical, Social worker groups
Natural park reserve	Firefighter, Law enforcement groups
School	Medical, Law enforcement,
	Firefighter, Social worker, Family groups

3.2 shows examples of grouping based on preferential interest. In what follows, we provide the assumptions and mechanism of our protocol to provide secure authenticated and confidential communication in application scenarios similar to the one described above.

Unlike other networks such as WSN and ad-hoc networks, our edge entity is not independent from the cloud; there need to be initial communications setup between the cloud and edge in order to manage the number of registered nodes such as patients and responsible personnel. In addition, all the information for involved citizens, such social insurance numbers, identities, and locations, are part of a trusted cloud system such as governmental institutes. Furthermore, all emergency incidents need to be reported to the main cloud to update the system's database. The presence of an edge entity here aims to off-load computation and mitigate communications delay to meet real-time requirements.

Problem Statement. The problems associated with our smart city application are similar to general Fog/Edge paradigms. Indeed, the nature of our case study of emergency and medical application requires meeting both high performance and security. In terms of performance, the challenging design issue is the low-end IoT devices associated with this network structure. While the edge entity is expected to have adequate resources and a sustainable power source, the streaming and receiver nodes, on the other hand, are assumed to have small computational power and low storage capacity (e.g., in the same range of Mica2 Mote [29], with 8MHz CPU and 4K RAM capacity). Thus, providing

secure broadcast communication which meets all real-time requirements between the edge entity and different groups requires special design. The conventional public cryptography approach provides a secure broadcast; however, it can be very expensive in terms of time and energy. The large number of computations will consume all the energy of the receivers.

In terms of security, the broadcast message for our application is intended to certain groups and may contain confidential data such as patients' information and locations. Furthermore, some information pertaining to the medical history of the patients including medical records and/or previous incidents may be also sent in these broadcasts. Thus, keeping confidential and private information of the involved participants is extremely important. Our design requires meeting both confidentiality and forward secrecy, especially because old-reported accidents may contain private medical records of a patient. In addition, if these broadcast messages are not authenticated, attackers can easily create false emergencies or deny the services of real ones.

3.4 Network Model, Security Assumptions, and Design Goals

In what follows, we explain the assumptions regarding our Fog/Edge computing IoT *(FEC-IoT)* and its network components.

3.4.1 Network Model

The edge network is composed of Access Points (APs) providing communication links to the edge IoT devices. Further, these APs are provisioned with computation resources to carry out the computation tasks of the edge devices. The radio access is assumed to follow any of the modern cellular radio access technology (e.g. New Radio or LTE). The aim of our protocol is to secure the downlink broadcast traffic that is sent from the edge entity to IoT devices. Our network model follows the three-layer hierarchy for

edge computing paradigm which is composed of cloud, edge, and IoT layers [134,162,179], also see Figure 3.1.

3.4.2 Security Assumptions and Adversarial Power

The edge entity is the center of information processing where it collects data from streaming sensor nodes. Then, based on this information, it sends messages to groups and communicates with receiver nodes. We assume the edge entity to be trusted. We also assume that each receiver node shares a permanent key and two seeds with the edge prior to deployment. We consider a probabilistic-polynomial-time (PPT) attacker having full control of the communication link between the edge entity and the group nodes such that the attacker can intervene in the message delivery or just eavesdrop on this communication. The attacker can also know when the message is received by the node and it can inject its own message or even tamper with the current message or replay old messages. Furthermore, we give the attacker additional power in knowing the secrets stored in the node's memory, indicating a compromised node, i.e., the attacker may take over a node (such as a bracelet worn by an authorized officer or a nurse in our case study) and know all its secrets. We also assume full synchronization between nodes and the edge entities in all their communications.

We assume static network for our smart city FEC-IoT application. There are several types of communication in our Fog/Edge paradigm, unicast, multicast, group broadcast, and general communication. The focus of this study is to secure group broadcast communication between the edge entity and receiver nodes. Specifically, we provide a hybrid hash-based keying protocol that secures the edge entity broadcast messages to group nodes.

3.4.3 Design Goals

The main security proprieties and performance goals provided in this design are:

- **Authentication, integrity, and confidentiality with limited resources:** Our proposal is a hybrid lightweight protocol that utilizes a one-way hash function to generate session keys. Confidentiality is provided from encryption while the authentication is provided from the OTS scheme used. The OTS scheme used is a one-way chain which provides both authentication and performance efficiencies.

- **Forward secrecy:** If any key is leaked, none of the previous communications should be exposed. To achieve this, we use hash based keying scheme to encrypt messages using temporary session keys which, if leaked, will not reveal previous secured communications. In addition, we do not deliver the session key using the permanent long-term key.

- **Efficiency:** Due to the limited computational power and limited storage capacity in FEC-IoT paradigm, secure protocols which provide authentication, confidentiality, and forward-secrecy must be light, efficient, and meet real-time requirements. Indeed, we use a hybrid hash based keying protocol with light computational and storage requirements that should support all types of such communications.

3.5 Protocol Specifications

The protocol is composed of three phases. The first phase is the initialization phase where the edge entity registers all active nodes into the system. The second phase is the session key agreement phase, and the third phase is the authenticated group broadcasting communication phase.

There are two types of keys used in this protocol, permanent and temporary. The use of temporary session keys provides confidentiality and forward secrecy which protects against compromising previously secured communications. The node's permanent key, on the other hand, is used in the initialization phase only to deliver the initial temporary keys. The initial temporary keys are destroyed and not used after the first communication

session, and the permanent key is used only in the initialization phase. In addition, two initial seeds are embedded in each node and edge entity prior to deployment.

Even though the edge entity is expected to have better resources, nevertheless, the protocol used for communication must be lightweight with limited communication bandwidth requirements to fit the requirements of the other side of the communication. These receiver nodes have various constraints such as under-powered processors, limited battery lifetime, small storage capacity, and limited bandwidth.

Our protocol clusters the network nodes into groups based on the preferential interest of end users. Each end user will have one or several groups associated to it, see examples shown in Table 3.2. Each group will have one temporary group key, and one temporary session key used during message delivery. It is reasonable to assume that the number of nodes per group is not fixed. Even though groups are isolated from one another, the edge entity must identify the nodes in each group and must have a shared permanent key with each node. Once all groups have been established, the edge entity communicates with individual groups using temporary group and session keys. The notations used in this proposed protocol are shown in Table 3.3.

Each node holds three keys, one permanent key and two temporary keys. Additionally, there are two seeds embedded in each node prior to deployment. The list of all keys is:

- **Individual keys:** Each node, u, in a given group shares an individual key with the edge entity, K_{eu}. These keys are set prior to deployment into the network. K_{eu} is a permanent key and is used to transfer the initial keys to each member of the group. In addition, K_{eu} can be used to deliver any other special instructions to individual nodes in the group.

- **Sub-session keys:** These keys, K_{egi}, are temporary keys, and they are part of a hash-based one-way chain [100], where i represents the index of the chain. The chain is set by the edge entity, and it is unique to each group. The commitment of this

Table 3.3: Notation used in FEC-IoT protocol specification and security analysis

Notation	**Description**
ID_u	Identity of node u
ID_{Group}	Identity of group
ID_e	Identity of edge entity
$\mathcal{A}$	Adversary
$\epsilon(.)$	Negligible function
N	Total number of nodes in the network
N_{Groups}	Total number of nodes in group
$\mathcal{F}_1$	Long-tem key (K_{eu}) reveal
$\mathcal{F}_2$	Session key (K_{Si}) reveal
$\mathcal{F}_3$	Ephemeral key ($K_{Ai}, K_{Bi}, K_{egi}, K_{Groupi}$) reveal
$\mathcal{R}(x)$	Random function
$\mathcal{E}_K$	Underlying encryption scheme with a key K
$\mathcal{M}_K$	MAC scheme with a key K
π	random permutation
T_i	Time of session
M	Number of messages
$\{Message\}_{Key}$	Authenticated encryption
M_i	Message at T_i
K_{eu}	Long-term key shared between edge and node
K_{egi}	Temporary key used to derive session key at T_i
K_{Ai}	Seed $K_{Ai} = H(K_{A(i-1)})$
K_{Bi}	Seed $K_{Bi} = H(K_{B(i-1)})$
K_{Si}	Session key at T_i
K_{Groupi}	Group key at T_i

chain, K_{eg0}, is delivered during the initialization phase to each node in the group using the permanent individual key, K_{eu}. The keys of the hash-based one-way chain are delivered to each group in reverse order in order to provide authentication of the edge messages to each node in the group. Each key in the sub-session key chain is used by the edge entity and group nodes to derive both the group and session keys. Each group has its own chain; thus, assuming l groups in the network, the

edge entity will carry l one-way chains.

- **Initial seeds:** Edge entity and all nodes will have two initial seeds pre-loaded prior to deployment, K_A and K_B. These seeds are used for generating both the session and group keys.

- **Group keys:** During each communication session, each group will have its own unique group key, K_{Group}, where the edge entity uses this key to encrypt the sub-session key and delivers it to nodes in the group. Furthermore, the edge entity and nodes derive the subsequent group key from the seed, K_{B_i}, and the current sub-session key, K_{egi} as

$$K_{Groupi} = H(K_{egi}|K_{B_i}), \tag{3.1}$$

- **Session keys:** Each message is encrypted with its own unique session key, K_{Si}. The i^{th} session key, K_{Si}, is derived from the seed, K_{A_i} as follows:

$$K_{Si} = H(K_{egi}|K_{A_i}) \tag{3.2}$$

Both the session and group keys are temporary, and they are both generated during the key agreement phase of our *FEC-IoT* protocol. The session key is used by the edge entity to encrypt the message while the group key is used by the edge entity to encrypt the sub-session key. Figure 3.3 shows the generation of the session and group keys on the one-way chain where this chain is produced by the edge entity and M represents the total number of messages. The sub-session keys, i.e., the one-way chain, is referred to as the authentication chain. Note that the authentication chain keys are delivered in the reverse order.

To broadcast a message to a particular group, the edge entity first delivers the message encrypted with the session key. For the node to derive this session key, it needs the

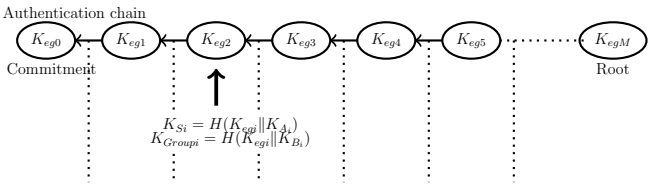

Figure 3.3: Session and Group Key Generation

current sub-session key in the chain. To this end, the edge entity delivers the sub-session key encrypted with the group key as shown in the following subsections. For example, if an elderly person fell the stairs, the edge entity starts a new communication with the medical group. The edge entity selects the next sub-session key in the authentication chain and delivers it to the medical group encrypted with its group key. In the same packet, the message also is encrypted with the session key. Subsequently, the nodes in the medical group derives the session key from the seed chain, K_A stored in the node, and the delivered sub-session key, K_{egi}, as shown in equations (3.1) and (3.2). In what follows, we give a detailed description of the three phases of the protocol.

3.5.1 Phase I: Initialization and grouping

Each node, u, has a shared permanent key with the edge entity and a unique identification, K_{eu} and ID_u, respectively. The edge entity communicates with nodes during this phase to establish groups and assigns each group a unique identification and the initial group key, K_{Group0}. It also creates a one-way sub-session key chain for each group and delivers the commitment of this chain, K_{eg0}, to all group nodes. There are two steps to the initialization phase as described below.

Finding active receiver nodes: After the cloud-edge setup, the edge broadcasts its, ID_e, to discover receiver nodes in the network. Then the edge entity broadcasts a HELLO message to everyone, and each active node responds to this call by providing the edge entity with its own identification, ID_u. This step is depicted in Figure 3.4 and outlined as

follows.

$$Edge \rightarrow * : ID_e$$
$$u \rightarrow Edge : ID_u, \{ID_u \| T_0\}_{K_{eu}}$$

where ID_e and ID_u are the identities of the edge entity and node, respectively. The encryption, $\{ID_u \| T_0\}_{K_{eu}}$, and henceforward in this chapter, represents an authenticated encryption. The edge entity receives responses from all nodes and verifies their identities as shown in Figure 3.4.

Assigning groups: In step 2 of the initialization phase, the edge entity divides nodes into groups based on the preferential interest of the end user. Prior to deployment, the edge entity has knowledge of all the end users' interests. The edge entity sends each node its initial group key and group identification, K_{Group0} and ID_{Group}, respectively. In addition, it sends the commitment of the sub-session key chain, K_{eg0}, to the group. The edge entity delivers the keys and IDs with authenticated encryption to individual nodes using the nodes permanent keys as shown below and depicted in Figure 3.4.

$$Edge \rightarrow u : ID_u, \ \{ID_{Group} \| K_{Group0} \| K_{eg0} \| T_0\}_{K_{eu}}$$

After this phase is finished, the individual key for each node, K_{eu}, is not used in message delivery.

3.5.2 Phase II: Key Agreement

The key agreement phase starts when the group nodes derive the keys needed to decrypt the message. The packet delivered from the edge entity contains the following two messages.

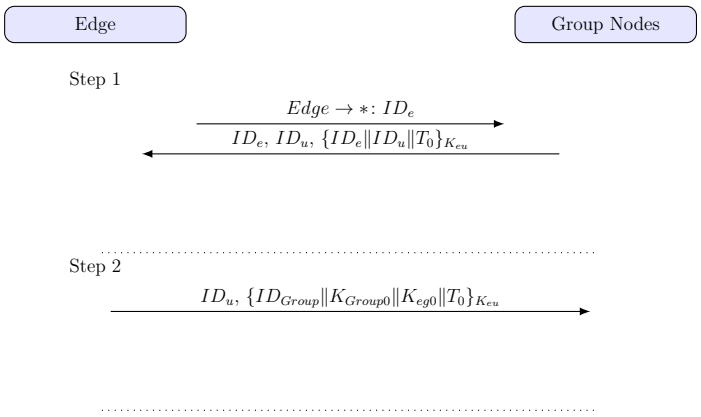

Figure 3.4: Phase I: Initialization and grouping

$$Edge \rightarrow group : ID_{Group}, \{M_i \| T_i\}_{K_{Si}}, \{K_{egi} \| T_i\}_{K_{Group(i-1)}}$$

The first part is the sub-session key chain, K_{egi}, encrypted with the current group key, $K_{Group(i-1)}$, where i represents the index of the communication session. It should be noted that the current group key is initialized prior to this communication session. Thus, the edge entity delivers the sub-session key, K_{egi}, encrypted with $K_{Group(i-1)}$. The second part of the message is the payload encrypted with the session key, K_{Si}. Each node derives the subsequent group key and the current session keys from Equations (3.1) and (3.2), respectively.

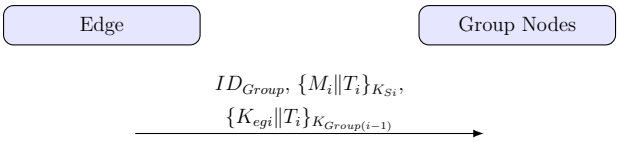

Figure 3.5: Phases II&III: Key Agreement and Authenticated Broadcasting

3.5.3 Phase III: Authenticated Broadcasting

In this phase, each node in the group decrypts the authenticated message. A given node first decrypts the sub-session key, K_{egi}, with the current group key, $K_{Group(i-1)}$. Then, the node verifies the authentication of the sub-session key by feeding it to a verify function given by:

$$VerifyEdge(K_{egi}, K_{eg(i-1)}) = \left\{ \begin{array}{ll} 1, & H(K_{egi}) = K_{eg(i-1)} \\ 0, & \text{Otherwise} \end{array} \right\} \tag{3.3}$$

The verification is simply done by hashing the received K_{egi} and comparing it to the previous stored sub-session key, $K_{eg(i-1)}$. If the verification passes, the node will generate the session key, K_{Si}, from equation (3.2) to decrypt the message. Subsequently, the node derives the next group key, K_{Groupi}, from equation (3.1) and saves it for the next communication session. Finally, the node erases the current session key, K_{Si}, and group key, $K_{Group(i-1)}$. The destroyed keys are no longer needed for future communications and keeping them may reveal a previously encrypted message if one node out of the group got completely comprised. Thus, we require each node to erase the previous group key and current session key after being used. The sequential steps followed for deriving and erasing the keys at a given period, T_i, are shown below:

1. Decrypt $\{K_{egi} \| T_i\}_{K_{Group(i-1)}}$ with $K_{Group(i-1)}$
2. $VerifyEdge(K_{egi}, K_{eg(i-1)}) \stackrel{?}{=} 1$
3. Derive $K_{Si} = H(K_{egi} \| K_{A_i})$
4. Decrypt $\{M_i \| T_i\}_{K_{Si}}$ with K_{Si}
5. Derive $K_{Groupi} = H(K_{egi} \| K_{B_i})$
6. $K_{A_{i+1}} = H(K_{A_i})$
7. $K_{B_{i+1}} = H(K_{B_i})$
8. Erase K_{Si} and $K_{Group(i-1)}$

The session key expires after this one-time use. To conduct another session, the edge entity again selects the next sub-session key from the authentication chain and uses it to encrypt the message. Then the edge entity repeats the process. Figure 3.6 shows a summary of the whole protocol indicating the authentication steps for each time slot, T_i. It should be noted that the sub-session key, K_{egi}, is a temporary key stored in the node for a one time period, T_i, and used for two purposes, authenticating the edge device and deriving the actual session key as shown in Equation (3.2).

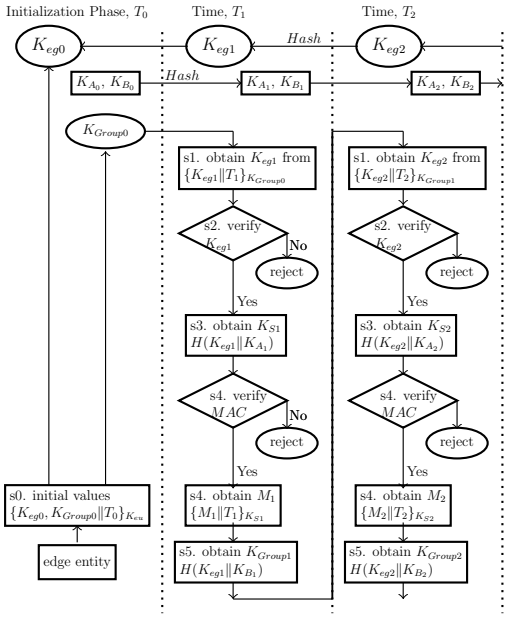

Figure 3.6: Summary of the proposed Protocol

3.6 Security Analysis

In what follows, we formally prove the security of the edge broadcast messages with respect to the claimed security goals, namely, confidentiality, integrity and forward secrecy. Without loss of generality, we assume that the *FEC-IoT* protocol implements an ideal authenticated encryption scheme using Encrypt-then-MAC where the underlying encryption scheme $\mathcal{E}$ is semantically secure under chosen plaintext attacks and the MAC scheme, $\mathcal{M}$, is unforgeable under chosen message attacks. Thus, both the generated

ciphertext and authentication tag are indifferentiable from that generated from a random permutation and random function, respectively.

Theorem 1 *The FEC-IoT protocol provides edge message broadcast integrity and confidentiality.*

Proof:

First, we start by proving the message integrity. Assume the existence of a PPT adversary $\mathcal{A}$ which is able to deliver a message such that an honest group node accepts it as an authenticated message from the edge entity. We build a distinguisher $\mathcal{D}$ which simulates the *FEC-IoT* protocol between the edge and nodes for $\mathcal{A}$. $\mathcal{D}$ queries a black-box function g with chosen inputs x, and is challenged to tell with non-negligible probability if $g(x) = \mathcal{M}_k(x)$, where $\mathcal{M}_k(.)$ is the MAC's psuodorandom function and k is a random secret key, or if $g(x) = \mathcal{R}(x)$, where $\mathcal{R}$ is a random function. For a security parameter n, the proof depicts the advantage of $\mathcal{A}$ in forging the MAC of a challenger message.

- $\mathcal{D}$ selects at random $b \in \{1 \dots M\}$ where M is the length of the edge authenticating chain

- $\mathcal{D}$ sends the initialization phase broadcast $\{K_{eg0}, K_{Group0} \| T_0\}_{K_{eu}}$ to $\mathcal{A}$, the regular broadcast *FEC-IoT* encrypted messages and tags, $\{M_i \| T_i\}_{K_{Si}}$, $\{K_{egi} \| T_i\}_{K_{Group(i-1)}}$, $1 < T_i \leq b-1$, where the authentication tag of a given message X is evaluated by $\mathcal{M}_{K_{Si}}(X)$

- At $T_i = b$, $\mathcal{D}$ sends to $\mathcal{A}$ a broadcast message with the exception that $\mathcal{D}$ replaces the tag which is originally evaluated by $\mathcal{M}_{K_{Sb}}(M_b \| T_b)$ with $g(M_b \| T_b)$.

- $\mathcal{D}$ inspects the message broadcast to group nodes and if $\mathcal{A}$ creates $\{M_b^{'} \| T_b\}_{K_{Sb}^{'}}$ where $M^{'} \neq M_b$ with a valid tag with respect to g, then $\mathcal{D}$, concludes that g is running $\mathcal{M}_k$, otherwise, g is running $\mathcal{R}$.

Let $\lambda(.)$ be a function such that the probability that $\mathcal{A}$ is able to forge the MAC, with length n, is given by:

$$\mathsf{Pr}[\mathcal{A}^{\mathcal{M}_k}(1^n) = 1] = \lambda(n).$$

Since it is always possible for $\mathcal{A}$ to succeed in forging the MAC with probability $\frac{1}{2^n}$ by producing the tag randomly, then the advantage of $\mathcal{A}$ is given by: $|\frac{1}{2^n} - \lambda(n)|$.

We will prove that $\lambda(n)$ is negligible (A function λ is negligible if for every polynomial $p(\cdot)$, there exists an N such that for all integers $n > N$, it holds that $\lambda(n) < \frac{1}{p(n)}$ [84]). Considering another MAC scheme running a truly random function, i.e., equivalent to g running $\mathcal{R}$, then $\mathcal{A}$ has only negligible probability to successfully forge any message, m, because the value $\mathcal{R}(m)$ is uniformly distributed in $\{0,1\}^n$ from the point of view of $\mathcal{A}$. Thus, we have

$$\mathsf{Pr}[\mathcal{A}^{\mathcal{R}}(1^n) = 1] \leq \frac{1}{2^n}.$$

It follows that if g is running $\mathcal{R}$ then $\mathcal{D}$ makes the wrong decision only with negligible probability. However, if the authentication is done using $\mathcal{M}_k$, then $\mathcal{A}$ forges with probability $\lambda(n)$, and $\mathcal{D}$ makes the right decision with probability $\frac{\lambda(n)}{b}$. Formally,

$$\mathsf{Pr}[\mathcal{D}^{\mathcal{M}_k}(1^n) = 1] = \frac{1}{b}\mathsf{Pr}[\mathcal{A}^{\mathcal{M}_k}(1^n) = 1] = \frac{\lambda(n)}{b},$$

and

$$\mathsf{Pr}[\mathcal{D}^{\mathcal{R}}(1^n) = 1] = \frac{1}{b}\mathsf{Pr}[\mathcal{A}^{\mathcal{R}}(1^n) = 1] \leq \frac{1}{b2^n},$$

Therefore, we have

$$|\mathsf{Pr}[\mathcal{D}^{\mathcal{M}_k}(1^n) = 1] - \mathsf{Pr}[\mathcal{D}^{\mathcal{R}}(1^n) = 1]| \geq \frac{1}{b}(\lambda(n) - \frac{1}{2^n}).$$

Under the semantic security assumption of the utilized MAC scheme where $\mathcal{M}_k$ is modelled as a pseudorandom function, $\frac{1}{b}(\lambda(n) - \frac{1}{2^n})$ must be negligible, and hence it follows that $\lambda(.)$ must be a negligible function. This implies that $\mathcal{A}$ succeeds in forging

the *FEC-IoT* broadcast messages with at most negligible probability and consequently the advantage of $\mathcal{A}$ is also negligible.

For proving message confidentiality, the above game is repeated. However, in this case, $\mathcal{A}$ is assumed to be able to decrypt ciphertexts generated by $\mathcal{E}_k$, and g is assumed to be running either $\mathcal{E}_k$ or π, where π is a random permutation. At the 3^{rd} step, $\mathcal{D}$ replaces the ciphertext which is originally evaluated by $\mathcal{E}_{K_{Sb}}(M_b \| T_b)$ with $g(M_b \| T_b)$, passes it to $\mathcal{A}$, and waits for the decrypted message, M'. If $M' \neq M_b$, then we conclude that g is running π, otherwise, g is running $\mathcal{E}_k$, both with block length m. Let $\gamma(.)$ be a function such that the probability that $\mathcal{A}$ is able to decrypt the ciphertext is given by:

$$\mathsf{Pr}[\mathcal{A}^{\mathcal{E}_k}(m) = 1] = \gamma(m).$$

Proceeding similar to the above proof of the integrity property, we get

$$|\mathsf{Pr}[\mathcal{D}^{\mathcal{E}_k}(1^m) = 1] - \mathsf{Pr}[\mathcal{D}^{\pi}(1^m) = 1]| \geq \frac{1}{b}(\gamma(m) - \frac{1}{2^m}).$$

Again, under the semantic security assumption of the utilized encryption scheme where $\mathcal{E}_k$ is modelled as pseudorandom permutation, $\frac{1}{b}(\gamma(m) - 2^{-m})$ must be negligible, and hence it follows that $\gamma(.)$ must be a negligible function. This implies that $\mathcal{A}$ succeeds in breaking the confidentiality of the *FEC-IoT* broadcast messages with at most negligible probability. Thus, the advantage of $\mathcal{A}$ is negligible.

■

Table 3.4: Adversarial Power

Query	**Keys Revealed**	**Adversary Gain**	**Security Breach**
Long-Term Key Reveal ($\mathcal{F}_1$)	K_{eu}	K_{eg0}, K_{Group0}	none
Session Key Reveal ($\mathcal{F}_2$)	K_{Si}	M_i	confidentiality and integrity of M_i
Ephemeral Key Reveal ($\mathcal{F}_3$)	K_A, K_B , K_{egi}, $K_{group(i-1)}$	M_i, M_{i+1}, M_{i+2}, ...	confidentiality of M_i, M_{i+1}, M_{i+2}, ...

Effect of adversarial powers: In addition to basic adversarial capabilities of the attacker, we let the attacker obtain secret information stored in the nodes memories via attacks modelled by three explicit queries [38]: *Long-Term Key Reveal*, *Session Key Reveal*, and *Ephemeral Key reveal* (see Table 3.4). In what follows, we show the effect of this adversarial powers on the view of $\mathcal{A}$ described in the proof of Theorem 1.

- **Revealing the long-term key.** By calling $\mathcal{F}_1$, K_{eu} is compromised which enables $\mathcal{A}$ to decrypt $\{K_{eg0}, K_{Group0} \| T_i\}_{K_{eu}}$. However, the compromise of K_{eu} does not enable $\mathcal{A}$ to derive the session key, K_{S1}, or the next group key, K_{Group1}, because both require the knowledge of the seed values, K_{Ai} and K_{Bi}, respectively. Thus, the view of $\mathcal{A}$ is the same as its view in the above game, and the proof of ensuring the message integrity and confidentiality proceeds in the same way.

- **Session key Reveal:** Calling $\mathcal{F}_2$ at T_i, enables $\mathcal{A}$ to recover K_{Si} and decrypt only $[\{M_i \| T_i\}_{K_{Si}}, \{K_{egi} \| T_i\}_{K_{Group(i-1)}}]$ which enables the attacker, $\mathcal{A}$, to reveal any message, M_i, during the selected session, T_i. However, the attacker is not able to derive the session and group keys at T_j where $i \neq j$. In what follows, we show that such an adversary has the same view as $\mathcal{A}$ during any session other than the i-th one.

  1. *Case $j > i$:* Assuming the use of one-way function for generating the authentication chain described in Section 3.5, the adversary, $\mathcal{A}$, cannot forge another message $[\{M_j \| T_j\}_{K_{Sj}}, \{K_{egj} \| T_j\}_{K_{Group(j-1)}}]$ where $j > i$ due to the lack of knowledge of K_{egj}, see Equations (3.1) and (3.2), respectively, and finding K_{egj} such that $K_{eg(j-1)} = H(K_{egj})$ has a negligible success probability.

  2. *Case $j < i$:* Similarly, $\mathcal{A}$, cannot derive previous session key, $K_{S(j-1)}$ (see Equation (3.2)) because this requires the knowledge of previous $K_{A_{(j-1)}}$ value. Since finding the previous $K_{A_{(j-1)}}$ such that $K_{A_j} = H(K_{A_{(j-1)}})$ contradicts the one-wayness assumption of hash functions, therefore, deriving the previous session key is infeasible.

- **Ephemeral key reveal:** By invoking $\mathcal{F}_3$ at T_i , $\mathcal{A}$ recovers K_{eu}, K_{Ai}, K_{Bi}, K_{egi}, and K_{Groupi} which compromise the security of all future messages because now $\mathcal{A}$ has the current seeds K_{Ai}, K_{Bi} which are required to generate future session and group keys through forward hashing. This function corresponds to a node's total compromise. However, at time $j < i$, under the assumption of the one waynerss of the employed hash function, the view of this adversary is the same as the view of $\mathcal{A}$ in the above game for all sessions at T_j. It is infeasible for $\mathcal{A}$ to evaluate K_{Aj} and K_{Bj} by evaluating the preimage of K_{Ai}, K_{Bi}. Accordingly, $\mathcal{A}$ cannot derive any previous session keys or group keys in order to break the confidentiality of earlier broadcast messages. Hence, we claim that *FEC-IoT* protocol provides forward secrecy.

3.7 Comparative Evaluation

In this section, we first present the computational, communication overhead, and memory requirements of our protocol, then we compare it to other lightweight protocols, specifically [1, 42, 60, 104, 106, 114, 140, 164, 179, 181, 206]. For our analysis, to maintain a security level of 128 *bits*, we assume that all keys used in this protocol including the long-term key, K_{eu}, and ephemeral session keys, i.e., K_{egi}, K_{A_i}, K_{B_i}, K_{Si}, and K_{Groupi} to be 128 *bits*. We assume that the time stamp is 32 bits and the message is of arbitrary length. The authenticated encryption MAC tag is of length 160 bits. Thus, the total extra bytes for delivering the sub-session key, $\{k_{egi} \| T_i\}_{K_{Group(i-1)}}$ = 128 bits for K_{egi} + 32 bits for T_i + 160 bits for MAC Tag = 320 bits = 40 Bytes. Additionally, there could be extra 16 Bytes from $\{M_i \| T_i\}_{K_{Si}}$, thus, the total communication overhead totals 56 Bytes.

As shown in Figure 3.6, our protocol is composed of eight steps, and in order to broadcast a message to a group, the edge entity is required to perform two encryption operations for encrypting the sub-session key and the message. Subsequently, upon receiving the broadcast message, the node is required to decrypt the sub-session key, message, and

derive group and session keys. Thus, in all steps of the protocol the node is required to perform two decryption operations; however, because of the timestamp overhead added to the message delivery, i.e. $\{M_i \| T_i\}_{K_{Si}}$, we consider the total number of decryption operations to be 3 for decrypting both the sub-session key and the message. In addition, we need 2 HMAC, and 5 hash-based computations for deriving keys and verification. Furthermore, each node is required to store the following keys, one permanent key with the edge entity, K_{eu}, two seeds, K_A & K_B, sub-session key, K_{eg}, two temporary session related keys, K_{Group} & K_S. Assuming 16 Bytes for each key, this totals to 96 Bytes. The edge entity, on the other hand, is required to store individual key for each node in the network along with the two seed values. In addition, the edge entity must generate a one-way chain to each group. Thus, the total storage space is $(2 + N + N_{Groups}) \times 16\ Bytes$, where N is the total number of nodes in the network and N_{Group} is the total number of groups.

Table 3.6 shows a performance comparison of our protocol with the protocols presented in [1,42,60,104,106,114,140,164,179,181,206]. As indicated in the table, our protocol requires less computational requirements and has less communication and storage overheads. In order to have a fair comparison between the studied protocols, we computed the execution times based on two different benchmarks, a high end Intel Core 2 Due CPU@2.4 GHz [179], and a low-end micro-controller ARM Cortex-M0 48MHz ATECC508A HW accelerated [186]. Our high-end benchmark is adopted from LAMANCO [179] while ARM Cotex-M0 is a low-end micro-controller of only 48MHz processor speed. However, both benchmarks do not include the operations used in [42, 106, 140, 164] of ID-Based pairing, ECC, and OTS signature verification; thus, for these protocols we list the execution times reported in the corresponding original publications. It should also be noted that both IMBAS and εIBAS are broadcast schemes where the computational delays are independent of the number of nodes. However, their listed computational run-time is, $6.6s$ and $3.4s$, respectively. The computational delays in IMBAS and εIBAS are high due to the large computational requirements for PKC ECC ID-Based pairing algorithms.

Table 3.5: Protocol Performance

Description	**Type**
Computational complexity-edge	2 Encryption
Computational complexity-node	3 Decrypt + 2 HMAC + 5 Hash
Communication overhead	56 *Bytes*
Storage requirements-edge	$(2 + N + N_{Groups})$
Storage requirements-node	96 *Bytes*

Table 3.6: Performance Comparisons

Ref	**Comm. Overhead**	**Storage Requirements**	**Computations**	**Benchmark** * *	**LAMANCO [179]** $^{\divideontimes}$ $^{\diamond}$
This work	56 *Bytes*	96 *Bytes*	3 Dec. + 2 HMAC + 5 Hash	$(3 \times 0.113 + 2 \times 0.722 + 5 \times 0.361) = 3.588$ *msec*	$(3 \times 107.4 + 2 \times 16.7 + 5 \times 6.0) = 385.6$ μsec
LAMANCO [179]	NA	55 *Bytes*	15 Hash+ 2 MAC	$(15 \times 0.361 + 2 \times 0.722) = 6.859$ *msec*	$(15 \times 6.0 + 2 \times 16.7) = 123.4$ μsec
LWMA [60]	DH Key Exch.	64 *Bytes*	DH+ HMAC	$(1 \times 134.2 + 1 \times 0.722) = 134.922$ *msec*	$(1 \times 33.49\ ms + 1 \times 16.7\ \mu sec) = 33.5$ *msec*
ULSS [1]	7.87 *KBytes*	32 *Bytes*	200 Hash+ 96 R-GEN	$(200 \times 0.361 + 96 \times 2.0) = 264.2$ *msec*	$(200 \times 6.0\ \mu sec + 96 \times 66.8 \mu sec) = 7.6$ *msec*
M-Tree [104]	112 *Bytes*	$\sim 2K$ *Bytes*	Dec. + Hash	$(1 \times 0.113 + 1 \times 0.361) = 0.474$ *msec*	$(1 \times 107.4 + 1 \times 6.0) = 113.4$ μsec
KMS-AMI [114]	NA	$(4 + N_{meters}, 4 + 2 \times N_{meters}) \sim 1K$ *Bytes*	2 HMAC	$(2 \times 0.722) = 1.444$ *msec*	$(2 \times 16.7) = 33.4$ μsec
LEAP [206]	NA	$(48 + 16 \times N_{neighbor})$ *Bytes*	MAC	$(1 \times 0.722) = 0.722$ *msec*	$(1 \times 16.7) = 16.7$ μsec
TSV [106]	80 *Bytes*	PK$\sim$ 10 *KBytes*	Signature verify	0.138 *sec* $^{\blacklozenge}$	
IMBAS [42]	198 *Bytes*	NA	ID-Based + pairing	6.6 *sec* $^{\blacklozenge}$	
εIBAS [164]	101 *Bytes*	NA	ID-Based+ pairing	3.4 *sec* $^{\blacklozenge}$	
2-Phase [140]	NA	1,530 *Bytes*	ECC-based cetificate	8.444 *sec* $^{\blacklozenge}$	
AKM-IoV [181]	128 *Bytes*	NA	13 Hash	$(13 \times 0.361) = 4.693$ *msec*	$(13 \times 6.0) = 78$ μsec

.
NA: Not available in the publication.
*: Based on ARM Cortex-M0 48MHz ATECC508A HW accelerated. AES 16 Bytes, Hash 64 Bytes, and RGN 64 Bytes.
*: Based on HMAC $\sim$ 2 Hash
$^{\divideontimes}$: Based on Intel Core 2 Duo CPU@2.4 GHz.
$^{\diamond}$: Based on RGN $\sim$ 4 HMAC [132]
$^{\blacklozenge}$: Execution times from original publication.

3.8 Summary

We proposed a protocol for smart medical emergency applications to be mounted on an edge device closer to the source of the data. All emergency incidents are addressed at the edge level leveraging the decentralized nature of edge computing. Our protocol provides edge broadcast message authenticity, confidentially, integrity, and forward secrecy. We have defined and proved the security properties of authentication and forward secrecy of our protocol. The computational complexity of our protocol is low and mostly based on hash functions and the required storage for each node is only 96 Bytes. On the other hand, the edge entity is required to store a permanent key for each node in the network,

and the chain of each group. The communication overhead is only 56 Bytes per session required for delivering the sub-session key. We compared our protocol to other lightweight broadcast protocols in terms of security and cryptography primitives. For future work, we will consider other efficiently computable and quantum resilient mechanisms to mitigate desynchronization issues. In other words, since both the edge and nodes have to maintain the same updated state in order to successfully verify the authenticity of received messages, a lost/dropped message interrupts this process and results in verification failure. However, if such a state can be derived from the encrypted message, then receiving nodes can self-synchronize. Accordingly, we plan to exploit the literature on stateless hash-based signatures to achieve that result in our future work.

Chapter 4

Lightweight Group Authentication Scheme

4.1 Introduction

Group authentication for the three-tier cloud-edge-IoT computing paradigm is gaining a lot of attention among the scientific research community. This is because massive machine type communication (mMTC) is an expected scenario in the EC smart applications, for example, a large number of IoT-nodes connecting to a single edge server, say thousands of nodes connecting to one edge computing server. In this scenario a de-centralized mass/group authentication is more efficient than authenticating one node at a time, because it reduces communication bandwidth on both authenticating server (AS) and IoT-nodes. Furthermore, in group authentication, each group member can authenticate other group members independent of the main edge server and without any certificates. However, the existing group authentication schemes 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 re-distributing new shares or re-running the set-up phase as described in [20, 51, 71, 105].

We propose a lightweight group authentication scheme based on secret-sharing, and symmetric-key cryptography. Our group authentication scheme at the edge ($GASE$) provides asynchronous multiple-authentication, key agreement, and forward secrecy. Also, it provides an efficient session key refreshing mechanism without re-distribution of the shares.

4.2 Related Work

In this section, we provide a brief summary of the development of group authentication schemes in the literature, and in a later section, we compare the most related research to our proposed protocol.

4.2.1 Group Authentication by Secret-sharing

Based on SSS, Harn proposed the first group authentication scheme in 2013 [71]. GAS allows group members to authenticate each other in a many-to-many group authentication style. In his paper, Harn presented three (n, t, m) group authentication schemes based on SSS: 1) basic, 2) asynchronous, and 3) asynchronous multiple authentications. 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. To be precise, 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, [20, 51, 105] propose their group authentication schemes based on Harn's scheme-1 or 2. For example, Aydin's *et al.* [20], use 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.* [105], utilize Harn's scheme-2 and ECC-pairing for group authentication and key agreement, respectively, for the LTE network. Similarly, Chien [51] uses Harn's scheme-2, ECC, and pairing for multiple-group authentications. However, these proposed GAS schemes [20, 51, 105], do not have any mechanism to refresh the secret session keys. Indeed, the GAS schemes proposed by [20, 51, 105] 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 group authentication scheme with multiple authentications 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.* [159], propose a fog-centred 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 *et al.* [87], scheme relies on Paillier threshold cryptography and requires public key cryptography (PKC) encryption and carries large computations. Similarly, Yang *et al.* [190] use bi-linear 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.

4.2.2 Group Authentication by Aggregated-MAC

Aggregated-MAC [83], 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, [97] propose a 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, [99] and [144] use the aggregated MAC for Third Generation Partnership Project (3GPP) and 6LoWPAN networks, respectively.

4.2.3 PUF-Based Group Authentication

Researchers utilized physical unclonable function devices for group authentication, group key agreement, and communication. For example, Yildiz *et al.* [196], propose a PLGAKD scheme which is based on PUF, CRT, and factorial tree security primitives to ensure group authentication and key distribution. On the other hand, [149] propose a group authentication scheme for the Narrow-Band IoT 5G framework which is simply based on PUF, hash, and xor functions. Other PUF-based proposals [58,75] are concerned with group communication and key distribution; however, they do not provide group authentication. A recent proposal by Chen *et al.* [48] utilizes PUF for mutual authentication leveraging Shamir secret sharing for availability and reliability. However, the scheme is not for group authentication or group communication.

4.2.4 Group Authentication Based on Multi-variate Polynomial

Other types of group authentication schemes proposed in the literature are based on symmetric multi-variate-polynomial security primitive [50, 68, 107]. Unlike the SSS

and Harn's schemes, the multi-variate polynomial is in the form of $f(x_1, x_2, \cdots, x_n)$ 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.* [107], propose a secret-sharing scheme based on bi-variate polynomial in which the scheme reduces the group manager computational complexity. However, the communication complexity remains the same as Harn's. Similarly, [50] propose a multi-variate-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.

4.2.5 Other Approaches Based on ECC

There are several proposals for massive machine-type communication device authentication in the literature. For example, Cao *et al.* [40], propose a massive narrow-band fast authentication scheme for the 3GPP 5G network. Their scheme is based on certificate-less aggregate *Signcryption* public key cryptography scheme. Similarly, [27, 98] use group authentication and key agreement based on ECC-cryptography and bi-linear pairing for the LTE and 5G framework, respectively, while [39] base their group authentication on Chebyshev-chaotic maps-based cryptography primitive. Chien [52] proposes 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 [33, 34, 47] are heavyweight with large overhead. Thus, we base our protocol on secret-sharing and aggregated MAC only.

Similarly, other group authentication schemes proposed in the literature are based on Chinese Reminder Theorem (CRT) [17, 54, 77]. However, because of the computational complexity and large communication overhead, we base our protocol on SSS instead of CRT. Table 4.1 shows summary of related protocols.

Finally, in terms of edge computing authentication protocols, only few proposals are found in the literature [76, 126, 156]. However, none of the cited proposals addresses

Table 4.1: *GASE* Related Protocols 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 [71]	– three GAS schemes – no key agreement	– group authentication – multiple authentication	– SSS scheme – exponential modular operation
Aydin [20]	– edge computing paradigm – GAS scheme – key agreement	– group authentication & confidentiality – no secret update mechanism	– SSS sharing – ECC cryptography
Li [105]	– LTE network – GAS scheme – key agreement	– group authentication & confidentiality, – single authentication	– Harn's scheme 2 – ECC cryptography – MAC
Chien [51]	– GAS scheme – no key agreement	– group authentication, – single authentication	– SSS scheme – bi-linear pairing
Shabisha [159]	– fog computing paradigm – group key agreement – pairwise fog-node key agreement	– group authentication & confidentiality, – single authentication	– ECC Cryptography – Schnorr signature – Lagrange interpolation
Yang [190]	– 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 [97]	– LTE network – massive machine type comm.	– massive MTC authentication – single authentication	– aggregated MAC
GLARM [99]	– 3GPP network – massive machine-2-machine comm.	– massive M2M authentication – single authentication	– aggregated MAC
Qiu [144]	– 6LoWPAN network – Proxy Mobile IPv6 – group mobility	– group authentication – group handover – single authentication	– aggregated MAC – encryption – hash + xor
PLGAKD [196]	– group comm. for IoT framework – smart lighting application – key distribution	– group authentication – node joining/revoking mechanism	– PUF-based devices – CRT + encryption – factorial tree + hash
Ren [149]	– 5G network – NB-IoT framework – key agreement	– massive authentication – secure data transmission	– PUF-based devices – Hash + XORs – truncated & aggregated authen. code
Li [107]	– GAS scheme – no key agreement	– group authentication & communication – single authentication	– SSS scheme – Bi-variate polynomial
Cheng [50]	– many-to-many group comm. – WSN paradigm – group key agreement	– membership authentication – group forward/backward secrecy – updated group key mechanism	– SSS scheme – multi-variate polynomial – hash
Cao [40]	– 5G network – massive narrow-band IoT (NB-IoT)	– access authentication & data transmission – secure data transfer	– encryption , – certificateless aggregate signcryption
SE-AKA [98]	– LTE network – group temporary key (GTK) agreement	– group authentication – single authentication	– ECDH exchange – MAC
LEGA [27]	– 5G network – mMTC	– group authentication & confidentiality – forward/backward secrecy – single authentication	– bi-linear pairing – aggregate certificateless signature mechanism
LSSA [39]	– 3GPP mobile devices – mMTC – key agreement	– access authentication – forward/backward secrecy – single authentication	– Chebyshev chaotic maps – encryption – MAC
Chien [52]	– group-oriented-range-bound – Authenticated-key-agreement	– aggregated & delegated authentication – homogeneous trust & authorization	– bilinear pairing – hash function

edge group authentication.

4.3 System Model and Design Objectives

4.3.1 System Model

The system model for our proposal is shown in Fig. 4.1. We have three network layers, the cloud layer, the edge or gateway layer, and the low-end IoT layer. We consider that the group leader and members communicate in a wireless medium in which each node, including the group leader, 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 group leader 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.

- **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.

- **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.

- **Group Leader** GL**:** The authentication server initially groups the registered nodes according to their geographical proximity, and for each group, the AS assigns a group leader. The group-leader selection could be based on different parameters such as available resources, distance from other nodes, communication range, or density [52, 89]. However, in our system the AS selects a group leader that has an adequate processing, power resources, and mobile broadband communication

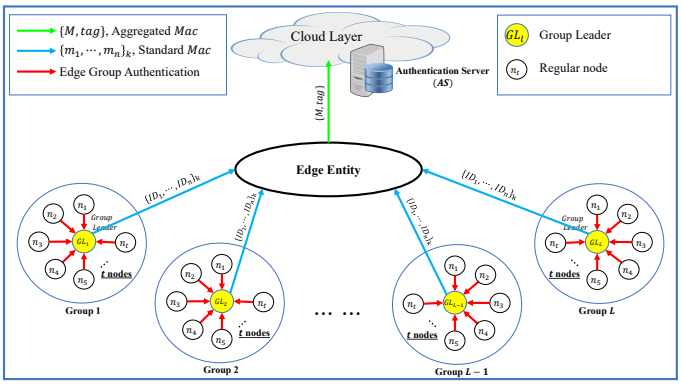

Figure 4.1: Group authentication in the three-tier cloud-edge-IoT framework

range. This is because the group leader 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 group leader needs to be available and active all the time. In case the group leader is inactive, the authentication process is not conducted. Alternatively, the authentication server may assign another back-up group leader such that if the main group leader is inactive, the back-up group leader conducts the authentication process.

- **IoT-nodes:** In each group, there are a 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 get encrypted with this key.

4.3.2 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 to 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 group leader and members is through wireless medium, the adversary may attempt several attacks such as eavesdropping, intercepting packets, re-directing packets, or injecting packets.

For our group setting, we consider two types of adversaries, an outside adversary, and 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 to 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 [71] describes 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 afterwords. 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 un-secured communications and conduct a replay attack in which the adversary re-run 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.

4.3.3 Protocol Goals

The goals of our GASE scheme are listed as follows.

1. **Group authentication and confidentiality:** The main objective of our proposal is to mass authenticate the IoT-nodes without congesting the authenticating server. Indeed, there could be a million 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 [71] in which the attacker waits for all participants to release their shares and releases its "good" share afterwords. Our goal is protecting 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 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 re-run 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 [37].

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 communication bandwidth.

The aforementioned features are not all included in other schemes found in the literature. Table 4.2 shows a comparison of the most relevant schemes [20, 51, 71, 105], where Harn-3 refers Harn's Scheme #3.

4.4 Edge Group Authentication Scheme

In this section, we present our lightweight group authentication scheme designed for the edge computing paradigm. Our protocol is composed of four phases, namely, initialization and setup phase, hashed-shares reveal phase, group leader authentication

Table 4.2: Comparison with relevant schemes

Parameters	**Harn-3 [71]**	**Aydin [20]**	**Li [105]**	**Chien [51]**	**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	✗	✗	✗	✗	✓

and secret recovery phase, and server authentication phase. The used notations are shown in Table 4.3.

4.4.1 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 2.3. 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 group leader (GL) as shown in Fig. 4.1. The authentication process occurs in three stages. First, the group leader authenticates all group members using Yang's multi-secret-sharing scheme, presented in Section 2.3. Then, the group leader sends the edge entity all group members identifications *IDs* authenticated with a standard *MAC*. Finally, the edge entity aggregates all tags received from each group leader and sends it to the authenticating server for verification.

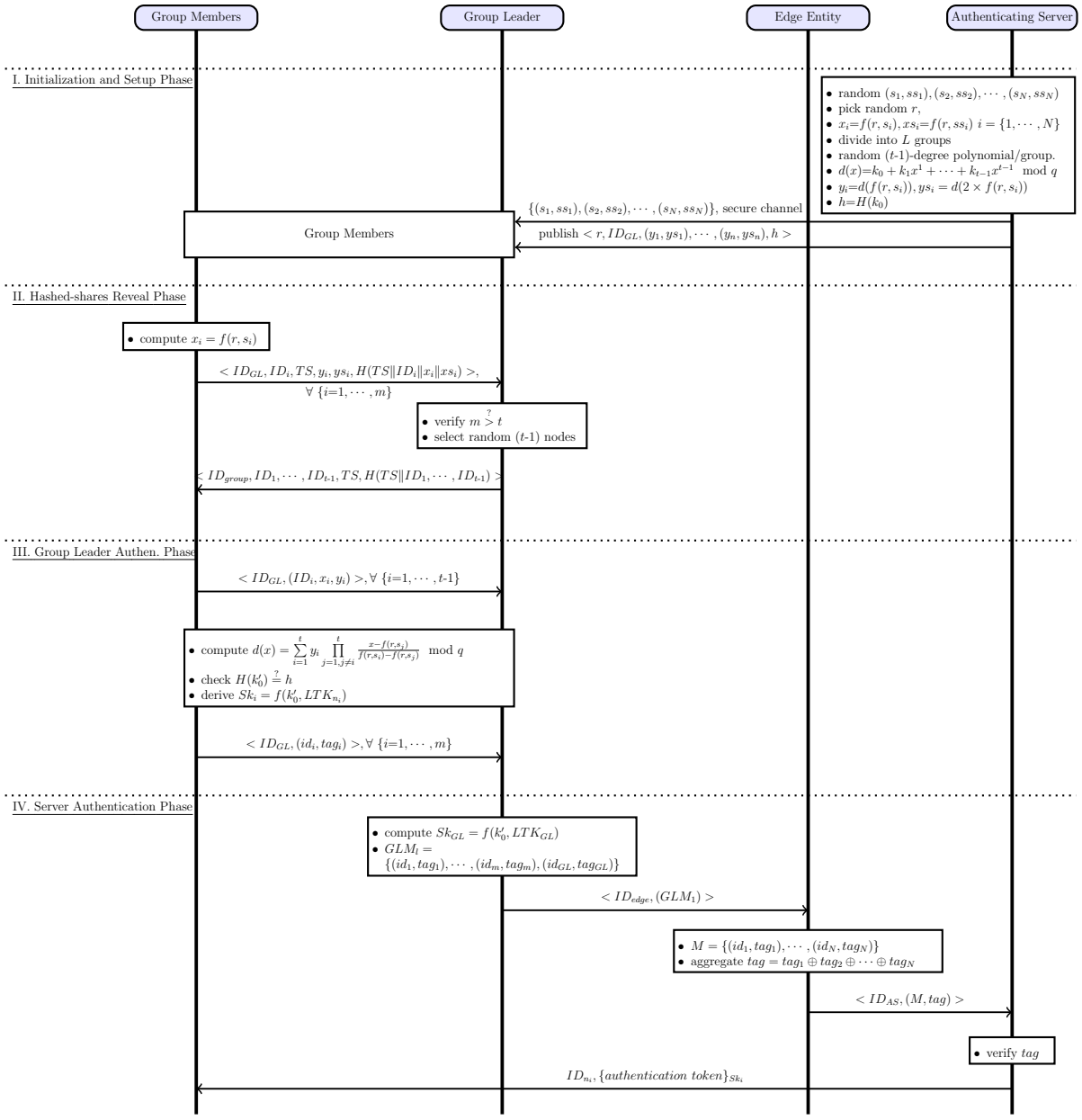

Figure 4.2: Multi-Authentication Group Authentication Scheme at the Edge.

Table 4.3: Notations used for *GASE*

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
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

4.4.2 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 equal $\geq t$, 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 [71]; the details of the asynchronous-release attack are shown in Section 4.5.2.

4.4.3 Initialization and Setup Phase

In our protocol, the authenticating server 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. 4.2.

a. AS randomly generates $2N$ secret-shadows, namely $(s_1, ss_1), (s_2, ss_2), \cdots, (s_N, ss_N)$, 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 4.4.2.

c. For each group, AS generates a unique (t-1)-degree polynomial in the format,

$$d(x) \quad = k_0 + k_1 x^1 + \cdots + k_{t-1} x^{t-1} \mod q \tag{4.1}$$

where q is a large prime and $0 < k_0, k_1, \cdots, k_{t-1} < q$, 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 $\{i = 1, \cdots, N\}$.

e. AS computes $h = H(k_0)$ where $H(\cdot)$ is a one-way hash function.

f. For each group, AS selects the group leader for the current session and publishes the following values.

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

4.4.4 Hashed-shares Reveal Phase

In this phase, the m participating nodes communicate with the group leader 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.

$node_i \rightarrow GL$:

$< ID_{GL}, ID_{group}, ID_i, TS, y_i, ys_i, H(TS \| ID_i \| x_i \| xs_i) >$

where $\{i = 1, \cdots, m\}$. The hash is used to protect against insider impersonation attacks as discussed in Section 4.5.

iii. After receiving all m participants hashed-shares, the group leader checks the number of participants. If

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

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

$GL \rightarrow ID_{group}$:
$< ID_1, \cdots, ID_{t\text{-}1}, TS, H(TS \| ID_1, \cdots, ID_{t\text{-}1}) >$

In this phase, it is important for each node to send its hashed-share in a given time window, w. Specifically, the group leader 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 4.5.

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 group leader aborts this current authentication session and requests the authentication server to assign lower degree polynomial which allows lesser number of nodes to conduct the authentication protocol.

4.4.5 Group Leader Authentication Phase

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

$node_i \rightarrow GL$: $< ID_{GL}, ID_{group}, (ID_i, x_i, y_i) >$

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

ii. Using Lagrange's interpolation formula, given in Equation (4.2), the group leader,

GL, and all other group nodes recover the polynomial and the secret k_0.

$$\begin{aligned} d(x) &= \sum_{i=1}^{t} y_i \prod_{j=1, j\neq i}^{t} \frac{x - f(r, s_j)}{f(r, s_i) - f(r, s_j)} \mod q \\ &= k_0 + k_1 x^1 + \cdots + k_{t-1} x^{t-1} \mod q \end{aligned} \tag{4.2}$$

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 is not sufficient to recover the (t-1)-degree polynomial secret.

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

$$H(k_0') \stackrel{?}{=} h \tag{4.3}$$

We note here that if Equation (4.3) holds, then all (t-1) nodes are authenticated by the group leader; otherwise, the entire (t-1) nodes are not authenticated. In the latter case, the group leader aborts the current authentication process and requests the authentication server to refresh the share's random number. Selecting a new set of nodes to reveal their shares; i.e. returning back to Step 4.4.4-iii., 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-t) nodes, these nodes deliver their secret shares using the recovered session key $\{x_i, x_{si}\}_{k_0}$. The group leader can verify the authenticity of these shares by checking them against their hashed values as described in Section 4.4.4-ii.

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

$$Sk_i = f(k'_0, LTK_{n_i}) \tag{4.4}$$

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 (id_i, tag_i) where the tag is a standard MAC using the node's session key, $tag_i = MAC_{SK_i}(id_i)$.

4.4.6 Server Authentication Phase

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

vi. The group leader, GL, combines all m group tags and appends its tag into, $GLM_l = \{(id_1, tag_1), \cdots, (id_m, tag_m), (id_{GL}, tag_{GL})\}$. GL sends this message to the edge entity. The key used for the GL-tag MAC is derived from the latest recovered secret, $Sk_{GL} = f(k'_0, LTK_{GL})$, where LTK_{GL} is the long-term key between the group leader and the AS.

$GL \rightarrow Edge$: $< ID_{edge},\ GLM_l >$

vii. The edge entity collects all group leaders' tags and aggregates them into one tag. Specifically, M=$\{(id_1, tag_1), (id_2, tag_2) \cdots, (id_N.tag_N)\}$ and tag=$tag_1 \oplus tag_2 \oplus \cdots \oplus tag_N$.

$Edge \rightarrow AS$: $< ID_{AS},\ (M, tag) >$

viii. The server receives the $Agg\text{-}MAC$ and verifies the tag using the new session key for each group leader.

4.4.7 Key Updates

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

$$< r_b, ID_{GLb}, (y_{(b,1)}, ys_{(b,1)}), \cdots, (y_{(b,n)}, ys_{(b,n)}), h_b >$$

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 re-distribution nodes secret-shares.

4.5 Security Analysis

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

4.5.1 Achieved Security Goals

We prove that our $GASE$ scheme is a secure secret-sharing scheme over $\mathbb{Z}_q$ according to Definition (7), [35].

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

In what follows, we show that the algorithm $\mathcal{G}$ which sends $(a_1, a_2, a_3, \cdots, a_{t-1}) \in \mathbb{Z}_q^{t-1}$ to $(y_1^{'}, \cdots, y_{t-1}^{'}) \in \mathbb{Z}_q^{t-1}$ which is the y-coordinates that their corresponding x-coordinates are $(x_1^{'}, \cdots, x_{t-1}^{'}) \in \mathbb{Z}_q^{t-1}$ is a one-to-one map.

We prove this by 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 $(x_1^{'}, \cdots, x_{t-1}^{'})$. 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, $(x_1, x_2, \cdots, x_t)$ and their corresponding values on the y-axis plane, $(y_1, y_2, \cdots, y_t)$, there exist only one unique (t-1)-degree polynomial that interpolates the data set $\{(x_1, y_1), \cdots, (x_t, y_t)\}$. 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 3 *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 (id_i, tag_i) message, $1 \leq i \leq m$ and replace it by (id'_i, 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 V-C-{a-f} for $\mathcal{A}$. Specifically, $\mathcal{D}$ selects a random (t-1)-degree polynomial, a random r, several m nodes, a group leader. We further let the long-term key LTK_{n_i} of the i-th 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 V-D-{i-iii} and Steps V-E{i-v}. In other words, at this stage of the protocol, the group leader (GL) authenticates all "selected" nodes.

3. In Step V-F{vi}, $\mathcal{D}$ simulates the transmission of all nodes IDs with their associated MAC to the group leader, i.e. $\{(id_1, tag_1), (id_2, tag_2), \cdots, (id_m, tag_m)\}$. However, for the i-th node, $1 \leq i \leq m$, $\mathcal{D}$ queries $\mathcal{C}$ with id_i and lets $(id_i, tag_i) = (id_i, \mathcal{O}(id_i))$. $\mathcal{D}$ hopes that $\mathcal{A}$ forges the authentication message of the i-th node.

4. $\mathcal{D}$ inspects the messages between $\mathcal{A}$ and AS. If $\mathcal{A}$ replaced the message (id_i, tag_i) with (id'_i, tag'_i) and AS authentication succeeds, then $\mathcal{D}$ concludes that the challenge $\mathcal{O}(id_i) = MAC_{SK_i}$; otherwise, $\mathcal{O}(id_i) = \mathcal{R}$.

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 i-th tag is given by

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

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

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

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 i-th message is $\frac{1}{m}\epsilon$, 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:

$$\mathsf{Pr}[\mathcal{D}^{MAC_k}(1^b) = 1] = \mathsf{Pr}[\mathcal{A}^{MAC_{ki}}(1^b) = 1] = \frac{1}{m}\delta(b),$$

and

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

Therefore, we have

$$|\mathsf{Pr}[\mathcal{D}^{MAC_k}(1^b) = 1] - \mathsf{Pr}[\mathcal{D}^{\mathcal{R}}(1^b) = 1]| \geq \frac{1}{m}\left(\delta(b) - \frac{1}{2^b}\right).$$

Under the semantic-security assumption that the MAC scheme is modelled using a pseudorandom function, See Definition (9), $\frac{1}{m}(\delta(b) - \frac{1}{2^b})$ must be negligible. It follows that $\delta(b)$ is negligible, which implies that $\mathcal{A}$ does not exist.

■

4.5.2 Other Attack Analysis

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

- **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.

  1. There are only (t-1) revealed shares in each session, and thus an outsider $\mathcal{A}$ knows only these revealed shares. Theorem 2 proves that our *GASE* protocol 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, $ID_i^{\mathcal{A}}$.

  2. 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 $GLM_l = \{(id_1, tag_1), \cdots, (id_m, tag_m), (id_{GL}, tag_{GL})\}$, as described in Step 4.4.6-vi. This requires the knowledge of the node's session key, SK_i, and it is infeasible to brute-force the group leader session key for key length ≥ 128 *bits*.

  3. 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 Equation (4.4). Thus, to obtain this key, $\mathcal{A}$ need the knowledge of k_0, LTK_i, and it is infeasible to brute-force this key for key length ≥ 128 *bits*.

- **Insider impersonation attack:** Similar to the outsider, 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, $\mathcal{A}$ pass

the authentication process and successfully retrieve the secret k_0 as described in Section 4.4. 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, inside attacker $\mathcal{A}$ has the same advantages as the outside attacker in impersonating a valid registered group member.

- **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 4.3-3. Our protocol is protected against this type of attack for the two types of adversaries.

  - The outside attacker: Only (t-1) shares are revealed in the authentication process, see Section 4.4.4. Thus, an outside attacker cannot recover the polynomial secrets with only (t-1) shares.

  - The inside attacker: All m participants must commit their two hashed-shares at the beginning of the authentication process, as see in Step 4.4.4-ii.; $< ID_i, TS, y_i, ys_i, H(TS \| ID_i \| x_i \| xs_i) >$. Since the group leader 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.

- **IoT-node collusion attack:** The group leader recovers the secret k_0 from substituting the group shares in the Lagrange's formula given in Equation (4.2). 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 $\lceil(t\text{-}2)/2\rceil$ 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 $\lfloor(t\text{-}1)/2\rfloor$ node collusion.

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

- **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 4.4.4-ii. The group leader does not accept any participants to join after the time window w elapses. In Theorem 2, 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.

- **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.

4.5.3 Analysis with Verifpal

Among formal protocol verification tools such as ProfVerif [31, 32, 32], Tamarin [146], AVISPA [18, 170, 177] EasyCrypt [24], and Verifpal [94], the latter has a built-in Shamir-secret-sharing primitive. Thus, we find it the most suitable one to model our protocol. In this subsection, we present a brief background on Verifpal, and we show our protocol's implementation and verification.

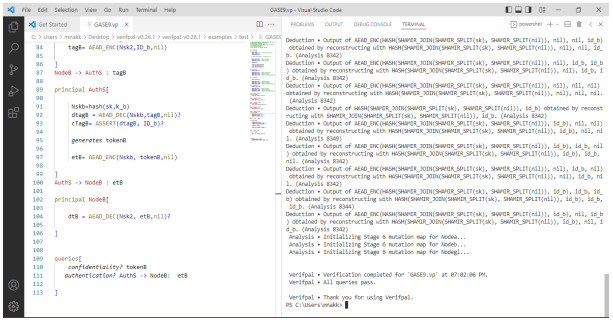

Figure 4.3: Verifpal Simulation Results

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 authentication server, namely, NodeA, NodeB, NodeGL, and AuthS, respectively. The authentication server 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 authentication server 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 authentication server. Fig. 4.3 shows the simulation results of *GASE* on Verifpal.

4.5.4 Notes on Untrusted GL/IoT-nodes Assumptions

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

- If a group leader 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" group leader and two types of

attacks, passive or active. In both passive and active attacks, the group leader cannot decrypt any secure communications between the nodes and the authentication server. This is because the group leader does not have 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, $Sk_i = f(k_o, LTK_{n_i})$, Equation (4.4). Thus, a compromised group leader cannot break the confidentiality of any node's message nor impersonate any group node. A compromised group leader 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.

- 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 authentication server.

- 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 4.3-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.

4.6 Comparative Evaluation

Similar to [155], we ran the following operations, $T_{mul,q}$,$T_{add,q}$, 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 [145]. Table 4.4 shows the operations simulated on Raspberry Pi-Model 4. We note here that we presented in Section 4.2 several group authentication protocols; however in the section we compare the most relevant schemes that are based on SSS with our scheme, specifically [20, 51, 71, 105], and present it in Table 4.5. We follow the theoretical analysis and notations given by Chein [51] to determine the computational complexity for users and the group leader 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, $GF(p)$ [71]. Also, Aydin *et al.* use $GF(q)$ operations as well as ECC multiplications and additions for authentication and key agreement.

Table 4.4: 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.

.

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 group leader for authentication. We assume that the two-variable one-way function is

Table 4.5: Performance Comparisons Group Authentication Phase

Ref	Comm (user)	Shares (user)	Computations/user	Complexity/ET$^{\blacklozenge}$	Key-agreement/ user
Proposed	$(t+m)$	2 *shadows*	$\lceil(2(m-2)+2)\times T_{mul,q}+1\times T_{inv,q}\rceil+(m-1)\times T_{add,q}+(m+2)\times hash$	$\cong (2m+238)T_{mul,q} \cong 2.7\ msec$ $^{\clubsuit}$	$=2\times hash\cong 5.8\mu sec$
Harn-3$^{\divideontimes}$ [71]	2×(m-1)	2 *shares*	$2\lceil(2(m-2)+3)\times T_{mul,q}+1\times T_{inv,q}\rceil+1\times T_{exp,p}+(m-1)\times mul_p+1\times hash$	$\cong (45m+1418)T_{mul,q} \cong 22\ msec$	NA *
Aydin-1 [20]	(m-1)	1 *share*	$1\times EC_{mul}$	$\cong (1189)T_{mul,q}\cong 12\ msec^{\star}$	(m-1)$\times ECDH+\lceil(2(m-2)+2)\times T_{mul,q}+T_{inv,q}+(m-1)\times T_{add,q}+1\times hash\rceil \cong 17.572\ msec$
Aydin-2 [20]	(m-1)	1 *share*	$\lceil(2(m-2)+1)\times T_{mul,q}+1\times T_{inv,q}\rceil+1\times EC_{mul}+(m-1)\times EC_{add}$	$\cong (7m+1421)T_{mul,q} \cong 15\ msec$ $^{\bigstar}$	$=(m-1)\times ECDH+\lceil(2(m-2)+2)\times T_{mul,q}+T_{inv,q}+(m-1)\times T_{add,q}+1\times hash\rceil \cong 17.572\ msec$
Li [105]	(m-1)	θ *shares*	$\theta\lceil(2(m-2)+3)\times T_{mul,q}+1\times T_{inv,q}\rceil+(m-1)\times T_{add,q}+1\times hash$	$\cong 41\theta(2m+239)T_{mul,q} \cong 222\ msec$	$1\times EC_{mul}+1\times hash+1\times MAC\cong 0.78\ msec$
Chien [51]	(m-1)	1 *share*	$\lceil(2(m-1)+2)\times T_{mul,q}+1\times T_{inv,q}\rceil+1\times EC_{mul}+m\times EC_{add}+2\times T_{pair}$	$\cong (7m+6785)T_{mul,q} \cong 67\ msec$	NA *

$^{\clubsuit}$: For case of GL authenticating rest of (m-t) nodes, we have extra encryption. Specifically, $ET\cong\lceil(2(m-2)+2)\times T_{mul,q}+1\times T_{inv,q}\rceil+(m-1)\times T_{add,q}+(m+2)\times hash+(m\text{-}t)\times T_{enc}\cong(2m+238)T_{mul,q}+(m\text{-}t)\times T_{enc}\cong 2.71\ msec$

$^{\divideontimes}$: Harn's scheme-3 [71].

*: Aydin case 1: Group manager confirming group members [20].

$^{\bigstar}$: Aydin case 2: Any group member confirming other group members [20].

$^{\blacklozenge}$: Based on Chien's approximation [51]; parameters are $t=10, m=20, \theta=2$.

$^{\blacklozenge}$: All execution times listed in Table IV are scaled to 32-byte data blocks.

*: not available in the scheme.

equivalent $\cong 1$ *hash* operation. For authentication, the group leader computes Lagrange's formula given in Equation (4.2). The group leader and all other members compute, $= 2\times hash+\lceil(2(m-2)+2)\times T_{mul,q}+1\times T_{inv,q}\rceil+(m-1)\times T_{add,q}$, where $T_{mul,q}, T_{inv,q}$ and $T_{add,q}$ are multiplication, multiplicative inverse, and addition in q, $GF(q)$, respectively. Finally, the group leader authenticates the recovered secrets by performing $(m+1)\times hash$ for hashing and verifying nodes original shares and *IDs*, see Equation (4.3) and Step 4.4.4-ii. For a fair comparison with [20, 51, 105], we follow Chien's approximation of ignoring hash and field-addition operations, and considering $T_{inv,q}\cong 240T_{mul,q}$. Thus, our approximated time complexity for the group leader is $\cong(2m-4+2+240)\times T_{mul,q}\cong(2m+238)T_{mul,q}$.

On the other hand, 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, case 1 and 2 in Table 4.5, 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 $=1\times EC_{mul}\cong 29T_{mul,P}\cong 29(41)T_{mul,q}\cong(1189)T_{mul,q}$. 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 $= \left[(2(m-2)+1) \times T_{mul,q} + 1 \times T_{inv,q} \right] + 1 \times EC_{mul} + (m-1) \times EC_{add} \cong (7m+1421) T_{mul,q}$ 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 4.5 shows comparisons between our scheme and other related schemes [20, 51, 71, 105] in terms of the computation complexity per user, approximated complexity per user, and group key agreement complexities. Fig. 4.4, 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, Adyin's scheme [20] have the group size 1-10, Li's scheme [105] have group size 1-180, and Chien's scheme [51] have group size 1-350. Nevertheless, our scheme can support larger group sizes. Our scheme has the least computations per user. Furthermore, our node key agreement requires the least amount of computations when compared to others; specifically, Aydin scheme requires $= (m-1) \times ECDH + \left[(2(m-2)+2) \times T_{mul,q} + T_{inv,q} + (m-1) \times T_{add,q} + 1 \times hash \right]$ per user and also Li's scheme requires $= 1 \times EC_{mul} + 1 \times hash + 1 \times MAC$ per user.

Communication per user: Unlike other schemes, we refresh secrets in each session. The AS, publishes a new set of public values, $(r, ID_{GL}, (y_1, ys_1), (y_2, ys_2), \cdots, (y_n, ys_n), h)$, 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, $Pkt_1 = < ID_i, TS, y_i, ys_i, H(TS \| ID_i \| x_i \| xs_i) >$, $Pkt_2 = < ID_1, \cdots, ID_{t\text{-}1}, TS, H(TS \| ID_1, \cdots, ID_{t\text{-}1}) >$, and $Pkt_3 = < (ID_i, x_i, y_i) >$. Assuming $TS = 4$ $Bytes$, $ID = 3$ $Bytes$, $SHA\text{-}256 = 32$ $Bytes$, $x_i = 32$ $Bytes$, and $y_i = 32$ $Bytes$, the total communication overhead per group is $((m + 2t - 2) \times 3$ $Bytes +$

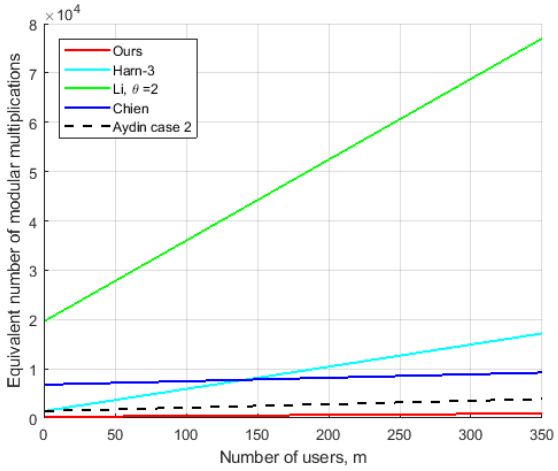

Figure 4.4: Number of multiplications per user for different schemes

$3 \times 4\ Bytes + 2t \times 32\ Bytes + (m+1) \times 32\ Bytes = (35m + 70t + 18)\ Bytes$. The communication overhead per user is $m + t$. Other schemes [20, 51, 71, 105], 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 $2 \times 32\ Bytes$. 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, Li's scheme also requires each node to store θ-polynomials secrets.

4.7 Summary

We proposed a lightweight group authentication scheme suitable for edge computing paradigm and massive node authentications. The protocol is based on Yang's multi-secret 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 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 run-time than others.

Chapter 5

Lightweight Physically Unclonable Function Group Authentication Scheme

5.1 Introduction

In this chapter, we extend our group authentication design leveraging Shamir's-secret-sharing scheme and low-cost physical primitives, PUFs, for flexibility and efficiency. Indeed, there are many limitations to the current proposed GAS schemes, and in this proposal we address these challenges as follows.

1. Most of the proposed GAS schemes in the literature are valid for one-time authentication [20, 71, 105]. For multiple-time authentications, it is required to have a new re-distribution of the shares in a secure environment. This may reduce the efficiency and flexibility of GAS schemes. To eliminate this, we use SSS-homomorphism property to support multiple-time group-authentications with the same set of shares.

2. Most of the GAS schemes require share storage at the nodes [20, 71, 105, 128]. The leakage of shares and node's secrets, say due to node's theft/capture, may jeopardize

the GAS security. To mitigate this threat, we utilize physically-unclonable-functions (PUFs) in our scheme such that the node's challenge-response-pair (CRP) is utilized to derive the node's secret-share instead of storing it.

3. Many GAS schemes proposed in the literature do not support key-agreement such as group session key [20, 71, 105, 128]. However, this is an important feature especially for some EC smart applications that require computations offload between group members. In our scheme, after the many-to-many group-authentication, we support establishing group session key as well as an individual session key for each node with the authenticating-server. This allows group members to offload tasks among them and individually communicate with the authenticating-server.

4. Most of the proposed GAS schemes do not support node-evicting/joining mechanism [20, 71, 105, 128]. However, in our proposal, using PUFs with SSS-homomorphism simplifies share-redistribution, node-joining, and node-evicting mechanisms.

5. Most of the proposed solutions using PUFs are vulnerable to modeling attacks [153]. In our proposal, we utilize the Shamir's SS $(+,+)$-homomorphic property to design an authentication protocol resistant to modeling attacks.

5.2 Related Work

To the best of our knowledge, PUFs are not utilized in the SSS GAS scheme in literature. Chen *et al.* [48] propose a PUF-based authentication scheme leveraging Shamir's-secret-sharing scheme; however, the proposal is for mutual authentication and not a group authentication. Also, their scheme requires large computations in the device and the authenticating server. Additionally, there are other group-authentication schemes using PUFs, but they are not based on secret sharing and require large amount of computations, e.g. [149, 188, 196]. Millwood *et al.* put forward a proposal for an Intra-

Group-based Authentication using DRAM-PUFs [122]. On the other hand, the literature has a plethora of PUF-based authentication schemes, e.g. [10, 23, 103, 117]. However, the scope of our work is group authentication using PUFs.

We note here that Chen *et al.* in [48] propose a PUF-based privacy-preserving authentication scheme which resists machine learning attacks. They employ a mechanism to randomly shuffle the mapping between challenges and responses in which the verification process is unaffected by the shuffling. However, their protocol is a mutual authentication scheme and not a group-authentication-scheme. Similarly, Mahalat *et al.* propose a mutual authentication scheme using PUFs and Pedersen's Verifiable secret-sharing (VSS) [117] for wireless sensors network. Also, Aman *et al.* propose a scalable authentication and privacy-preserving scheme using PUFs and special layered architecture for the Internet of vehicles [10]. However, these solutions are not GAS schemes. Finally, Lee *et al.* propose a group-authentication key-agreement scheme; however, their design requires a special Barrel-shifter PUF [103].

In the literature, we see edge-computing applications concerned with grouping and clustering which require authentication. For example, Aydin *et al.* [21] propose a group-authentication-scheme for swarm-based authentication as a direct application of their group-authentication-scheme proposed in [20]. On the other hand, for a group-oriented-range-bound, Chien [52] puts forward a delegated authentication-key-agreement (AKE) scheme which uses bi-linear pairing PKC and hash functions. In [23], Banasal *et al.* put forward a scheme for UAVs that utilizes SSS and PUFs, however, their scheme is a mutual authentication and not a group-authentication. Similarly, Pu *et al.* propose a mutual authentication, privacy-preserving, and key-agreement utilizing PUFs and chaotic systems for Internet of drones [142]. A scalable authentication location-aware clustering for the Internet of drones is proposed in [22]. Similarly, Yu *et al.* in [193] propose a lightweight PUF-based scheme for the Internet of drones in smart cities. Additionally, lightweight solutions are proposed in the literature for edge-computing broadcast and

Table 5.1: *PUF-GASE* Related Protocols Summary

Reference	Application	Security Primitives	Advantages	Disadvantages
This work	– GAS – edge-computing – de-centralized	– SSS – SSS-homorphism – PUF – hash & xor	– lightweight – group authen. – integrity, FWD secrecy – multiple-time authen.	– PUF sensitivity – requires PUF-FE
Harn-3 [71]	– GAS – de-centralized	– SSS – SSS-homorphism – mod-exponentiation	– group authen. – multiple-time authen.	– heavyweight – no key-agreement – no share-update
Li [105]	– GAS – LTE – MTC	– SSS – ECC – bi-linear pairing	– group authen. – key-agreement	– no share-update – one-time authen.
Aydin [20]	– GAS – IoT – de-centralized	– SSS - ECC – ECDH	– group authen. – key-agreement – integrity	– no share-update – one-time authen.
GASE [128]	– GAS – edge-computing – de-centralized	– SSS – MAC	– lightweight – group authen. – integrity	– no node revoking mechanism
Chien [51]	– GAS	– SSS – ECC	– group authen. – multiple-time authen.	– no key-agreement
Kaya [87]	– threshold Cryp.	– Asmuth–Bloom SS – Pailliar threshold	– group authen.	– heavyweight
Shabisha [159]	– fog-computing	– Schnorr signature – ECC	– group authen. – key-agreement	– heavyweight
Yildiz [196]	– IoT – smart lighting – de-centralized	– PUF – CRT – factorial-tree	– group authen. – node evicting	– large overhead – PUF sensitivity
Xia [188]	– smart home – IoT-applications – VANET	– PUF – CRT – multi-device authen.	– group authen. – key-agreement – dynamic join/leave	– PUF sensitivity – latge overhead – no share-update
Ren [149]	– Narrowband-IoT – data-transmission – 3GPP 5G	– PUF – Aggregated MAC	– group authen. – mutual authen. – key-agreement	– PUF sensitivity – one-time authen.
Lee [103]	– Medical IoT – sensors	– Barrel-Shifter PUF – SHA256	– dynamic group authen. – key-agreement – anonymity	– PUF sensitivity – Specific to BS-PUFs
Chien [52]	– IoT – 5G, LTE,& UMTS	– Bilinear pairing – BDHP & BPI	– group authen. – key-agreement	– heavyweight
Millwood [122]	– modified CNN – low-end devices	– DRAM PUF – PUF-Phenotype	– group PUF authen. – no helper-data	– specific to DRAM

others [126, 127, 155], but not for group-authentication.

Table 5.1 shows comparisons of all related authentication schemes presented in this section. The table compares authentication protocols in terms of applications, utilized security primitives, advantages, and their disadvantages.

5.3 Applications, Security Issues, and Design Goals

5.3.1 Applications

Our new *PUF-GAS* scheme is suitable for an edge-computing grouping scenario that requires low run-time, small communication overhead, and high security levels. Indeed, it is a common case in the three-tier cloud-edge-IoT smart applications to have low-end IoT-devices that cannot support heavyweight computations. The IoT-devices maybe equipped with under-powered processors in the range of few MHz, have low battery-

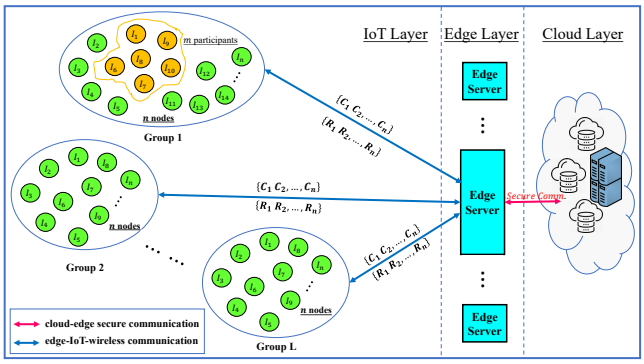

Figure 5.1: PUF-based SSS Group Authentication Scheme

lifetime, and have small storage capacity [88]. It is also a common case in the edge-computing paradigm to have several nodes attached to one edge-server. Authenticating all of these nodes at the edge of the network is more efficient than authenticating one node at a time from the cloud server. In this work, we propose a scheme that supports group-authentication and allows group-key-agreement. The general scenario for our grouping system is shown in Fig. 5.1.

As an example, our *PUF-GAS* scheme is suitable for a swarm-based unmanned aerial vehicles (UAVs) system authentication. The UAV system could be composed of a main-server (MS), base stations, and several UAVs/drones. All registered drones in the system initially get divided by the main-server into swarms/fleets according to their capacity and tasks. Thereafter, the fleets get deployed to execute their specific missions and may enter into one or more registered base station zones. Each fleet flying into a registered base station zone gets authenticated, recovers a group key, and establishes individual session keys with the base station edge-server. Consequently, the authenticated fleet-drones communicate with each other and with the base station without referring to the main-server.

5.3.2 Security Issues: Assumptions and Attack Model

We assume that the main-server and the edge-server are trusted entities, and all communications between them are also secure. On the other hand, the group nodes are accessible to the attacker and vulnerable to capture or physical theft. To mitigate this vulnerability, we require all group-nodes to have PUFs such that the shares are not stored in the nodes; instead, shares are derived from their PUFs. Thus, even if a node got captured, it does not leak its shares. The communications between the edge-server and group nodes are not secure and subject to eavesdropping. For our system, we consider the following threats.

- *Node-impersonation attack:* In this type of attack, the adversary tries to forge a communication between a valid group-member and an edge-server. We show in Section 5.6 that the adversary's advantage in this attack is equivalent to that of a brute-force attack.

- *Node-collusion:* Our (t, n)-secret-sharing scheme is resistant against (t-1) node collusion. This is because in order to recover the (t-1)-degree polynomial, the attacker must corrupt t valid nodes and retrieve their shares.

- *Asynchronous-share-release:* For a (t-1)-degree polynomial, an adversary who does not have a valid share waits for all t participants to reveal their shares and then recovers the polynomial and generates a valid share. In our scheme, this threat does not exist, because the group nodes never reveal any of their shares.

- *Node-capture/theft:* In a typical node theft, the attacker physically accesses the node and retrieves all its secrets. However, in our scheme, shares are not stored, instead they are derived from the nodes PUF-devices. Thus, the adversary who captures a node cannot leak its shares or any other node's share.

- *Replay-attack:* Our scheme is protected against reply attacks, because we always

ensure freshness in both authentication sessions and group key recovery sessions.

5.3.3 Design Goals

The goals of our scheme are as follows.

1. *Multiple-time group authentication and key-agreement:* The primary advantage of our protocol is the multiple-time group-authentication. Specifically, the typical PUF protocols require the server to store a large number of CRPs for each node in the system, and each CRP is used for only one authentication. On the other hand, our scheme allows for multiple-time group-authentication for the same set of CRPs. After each authentication session, the edge-server and the group-nodes derive group-session key which allows all parties to communicate with each other independently from the main-server.

2. *Node-evicting mechanism and flexibility:* The second primary advantage of our protocol is the simple node-evicting mechanism. Indeed, if one or more nodes got captured, the edge-server manages to authenticate the other group-nodes without re-initialization or new share re-distribution. The node-evicting mechanism is explained in details in Sections 5.4 and 5.5. Additionally, the authentication process is flexible to allow any m group-nodes to participate in the authentication process provided that $m \geq t$.

3. *Lightweight and secure storage:* The security primitives used in our protocol are simple and lightweight which make them suitable for the low-end devices. Specifically, we use SSS and hash functions. Also, using PUFs mitigates any insecure storage of shares inside the nodes.

4. *Forward and backward secrecy:* The breach or leakage of a session key should not affect previous or future session keys, and this defines forward and backward secrecy,

respectively. Specifically, if an adversary somehow leaks a group-key associated with a session x, previous (x-j) and future (x+j) group-session keys are not leaked. This mitigates the adversary's advantages in breaching the confidentiality of encrypted messages. In our protocol, we support forward and backward secrecy. Specifically, our *PUF-GAS* protocol provides session key-agreements that are independent of future and previous sessions.

5.4 Constructing SSS Using PUF

Our main goal for using PUFs with Shamir's secret-sharing-scheme is to mitigate possible attacks resulting from insecure storage. The straightforward solution for applying PUFs in SSS for a group of n nodes is to have each node's CRP act as its hidden share then construct a polynomial from the n CRPs. While this solution assures no-share storage at the node, it breaks the definition of threshold sharing-scheme. Specifically, the constructed polynomial for the straight-forward solution must be fixed to (n-1)-degree, and this requires all group members to participate in the authentication process; thus, any subset of the shares does not recover the secret. In what follows, we show a construction of a SSS using PUFs which achieves multiple-time asynchronous group-authentication without restricting the number of participants.

5.4.1 Assumption

We assume that the edge-server (ES) authenticates a group of n nodes on behalf of the distant main-server (MS). We also assume that each node is equipped with a PUF and derives its own unique challenge. The PUF-response to this unique challenge is the member's SSS share in which no storage of the secret-share is required. The underlying security property of PUFs, see Definition (10), states that it is infeasible for any adversary to generate $R_i' = R_i$ without having the physical device. Thus, it is infeasible for an

attacker to generate a valid share and impersonate a node.

5.4.2 Construction

In what follows, we show the construction of K SSS-polynomials using PUFs, and Fig. 5.2 shows an illustrative example of $K = 5$ and a cubic-polynomial formation.

1. **Collecting CRPs:**

   (a) The main-server sends K challenges to the group of n nodes, in which each node is equipped with a PUF. Each node derives its PUF-responses and its helper-data, as shown in Table 5.2, and sends them back in a secure channel.

   (b) The main-server receives
   $\{(C_1, R_{(1,1)}, hd_{(1,1)}), \cdots, (C_K, R_{(1,K)}, hd_{(1,K)})\}, \cdots,$
   $\{(C_1, R_{(n,1)}, hd_{(n,1)}), \cdots, (C_K, R_{(n,K)}, hd_{(n,K)})\}$

2. **Polynomial constructions:**

   (a) For each node, the main-server has K different CRPs $(C_k, R_{(i,k)})$, and translates each CRP to an (x, y)-point in the xy-plane. The main-server stores the helper data for each node.

   (b) The main-server creates upto K polynomials. The degrees of the polynomials varies from n-degree to (t-1)-degree polynomials where $t < n$.

   (c) Fig. 5.2 shows an example of $K = 5$ and the construction of a cubic-polynomial, e.g. degree = 3. In the example, the polynomial has 4 xy-points laying on the curve and 1 xy-point laying outside the curve.

3. **Storage and delivery:**

   (a) For the (n-t) xy-points laying outside the curve, the offsets of these points are stored along with the polynomial.

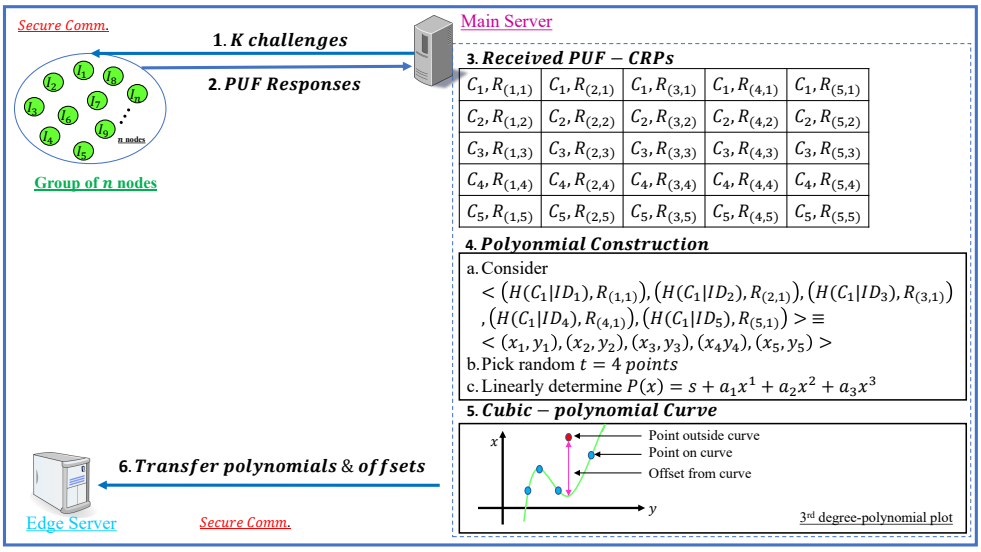

Figure 5.2: Example of PUF SSS Polynomial Construction, $K = 5, t = 4$, *Cubic-polynomial*

(b) The main-server sends a copy of the K polynomials, the helper data, and the offsets to the edge-server in a secure channel.

Table 5.2: Node-responses for different challenges

Challenge	Node i response for k^{th} challenge	helper-data
1	$R_{(i,1)} = PUF(C_1)$	$hd_{(i,1)}$
2	$R_{(i,2)} = PUF(C_2)$	$hd_{(i,2)}$
⋮	⋮	⋮
K	$R_{(i,K)} = PUF(C_K)$	$hd_{(i,K)}$

5.4.3 Node-evicting mechanism

Finally, the mechanism of evicting a group-member is done locally at the edge-server, and without any share-redistribution from the main server. Simply in this case, the edge-server creates a new (t-1)-polynomial in which the evicted member does not have any shares, and the rest of the nodes are not affected. Similarly, updating the group shares requires only changing the group challenge.

5.5 Proposed Scheme Specifications

In what follows, we list the used notations in Table 5.3 and then give the details of the *PUF-GAS* scheme shown in Fig. 5.3.

Table 5.3: Notations used for *PUF-GAS*

Notation	**Description**
N	The total number of nodes in the system
K	Total number of polynomials per group
k	Polynomial/challenge index #
L	Total number of groups
l	Group index
m	Number of participating nodes in the group
n	Number of nodes in the group
t	Threshold of SSS-polynomial for the group
b	The secondary polynomial index and number of hashes
ID_i	Node i identification
C_k	Group Challenge for session k
C_i	Challenge of node i, $C_i = H(ID_i \| C_k)$
$P_{(k,l)}(x)$	Polynomial k of the l-th group
$P_{(k,l,b)}(x)$	Secondary b polynomial k of the l-th group
$R_{(i,k)}$	PUF response of node i of k^{th} polynomial
n_a	Nonce
Tkn	Session token for the group
SK_i	Session key for node i
$GK_{(k,b)}$	Group key for authentication session k
$H(.)$	One-way hash function

5.5.1 Overview

We use PUFs with SSS and its $(+,+)$-homomorphism property for K authentication sessions as follows.

- Initially, the main-server (MS) divides the total number of registered nodes, N, into L

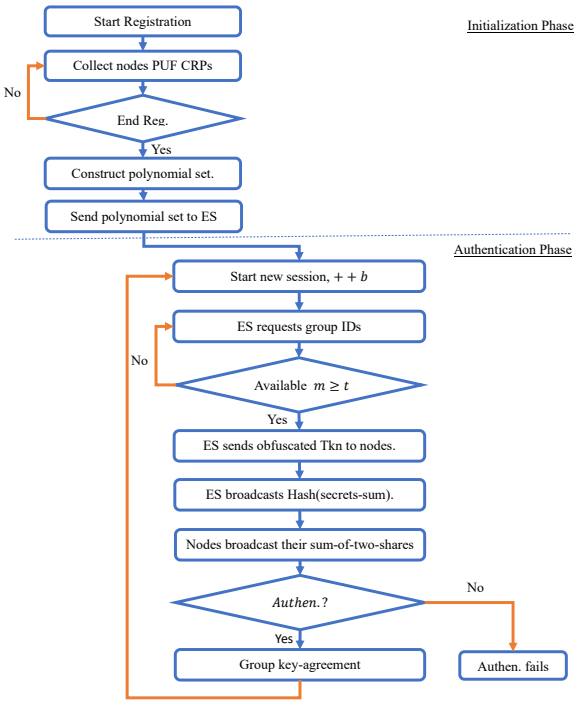

Figure 5.3: Flow diagram of *PUF-GAS*

different groups based on given criteria such as resources, capacities, or missions. To build the polynomial reservoir, the main-server creates K unique polynomials for each group.

- Each group has K polynomials, and each polynomial is constructed from the CRPs to its k^{th} group challenge, C_k where $k \in \{1, \cdots, K\}$. Specifically, the CRP is $(C_{(i,k)}, PUF(C_{(i,k)}))$, respectively for each node in the group, where $C_{(i,k)} = H(ID_i \| C_k)$, $H(.)$ is a one-way hash function, and i is the node index. From the collection of n CRPs, the main-server selects a (t-1)-degree polynomial, $P_{(k,l)}(x)$, for each authentication-session k, where the threshold is $t \leq n$.

- The total number of polynomials for the system is $K \times L$, and after building the reservoir, the main-server transfers the entire polynomials to the edge-server via a secure channel. These polynomials are used for de-centralized authentication at the edge.

- In order to achieve multiple-time authentication, we utilize the SSS $(+,+)$-homomorphic property. Specifically, when the edge-server needs to authenticate a group, it uses one of the K polynomials and derives its secondary polynomial. The secondary polynomial is constructed from the available $(m \geq t)$ participating nodes and has the same x-axis points as the original group polynomial; thus, the secondary-CRP is $(C_{(i,k)}, H^b(PUF(C_{(i,k)})))$, respectively for each node in the group, where b is the number of hashes, and the secondary polynomial index.

- Finally, the shares for the two polynomials, $P_{(k,l)}(x)$ and $P_{(k,l,b)}(x)$, are never revealed. Instead, each node sends its sum-of-shares xor-ed with a group token $H^b(Tkn)$. Thus, the total number of authentications per group is $K \times b$.

The steps are shown in the following two phases, namely, the registration and initialization phase and the group-authentication and key-agreement phase.

5.5.2 Registration & Initialization Phase

The main-server first divides the registered nodes into L groups and sets up the polynomial reservoir.

- **Step I:**

  a. The main-server sends K unique challenges to the registered nodes in the system. Each k^{th}-challenge corresponds to the k^{th}-polynomial.

  $$MS \rightarrow Nodes\colon \{(C_1, \cdots, C_K)\}$$

  b. Each node i derives its unique PUF challenge, $C_{(i,k)} = H(ID_i \| C_k)$ for each $k \in \{1, \cdots, K\}$ respectively, and then obtains the corresponding PUF-responses as shown in Equation (5.1).

$$R_{(i,k)} \quad = PUF(C_i) = PUF(H(ID_i \| C_k)) \tag{5.1}$$

c. Each node i sends its PUF-responses, see Table 5.2, back to the main server via a secure channel.

$Node_i \rightarrow MS$: $\{ID_i, R_{(i,1)}, \cdots, R_{(i,K)}\}$

- **Step II:**

The main server constructs K (t-1)-degree polynomials for each group as follows.

1. To construct a single polynomial for a group, the main-server randomly selects t out of the $\{(x_1, y_1), (x_2, y_2) \cdots, (x_n, y_n)\}$ points. The (t-1)-degree-polynomial is in the format of Equation (5.2).
2. To compute the polynomial's coefficients $\{a_{1(k,l)}, \ldots, a_{t-1(k,l)}\}$, the main-server uses the linear $\mathbf{a} = (\mathbf{x^T}.\mathbf{x})^{-\mathbf{1}}\mathbf{x^T}.\mathbf{y}$ where $\mathbf{x}$ is the Vandermonde matrix of x-points, $\mathbf{y}$ is the vector of y-points, and $\mathbf{a}$ is the vector of the polynomial's coefficients.
3. To reserve the rest of the (n-t)-points, which have their (x, y)-points laying outside the polynomial-curve, the main-server stores them. This is because, in the authentication phase, any of the n points can participate in the authentication process regardless of the selected t-points for the polynomial $P_{(k,l)}(x)$.

$$\begin{aligned} P_{(k,l)}(x) \quad &= s_{a_{(k,l)}} + a1_{(k,l)}x^1 + \cdots \\ &\quad + a_{t\text{-}1_{(k,l)}}x^{t\text{-}1} \mod q \end{aligned} \tag{5.2}$$

where k is the polynomial/challenge index and l is the group index.

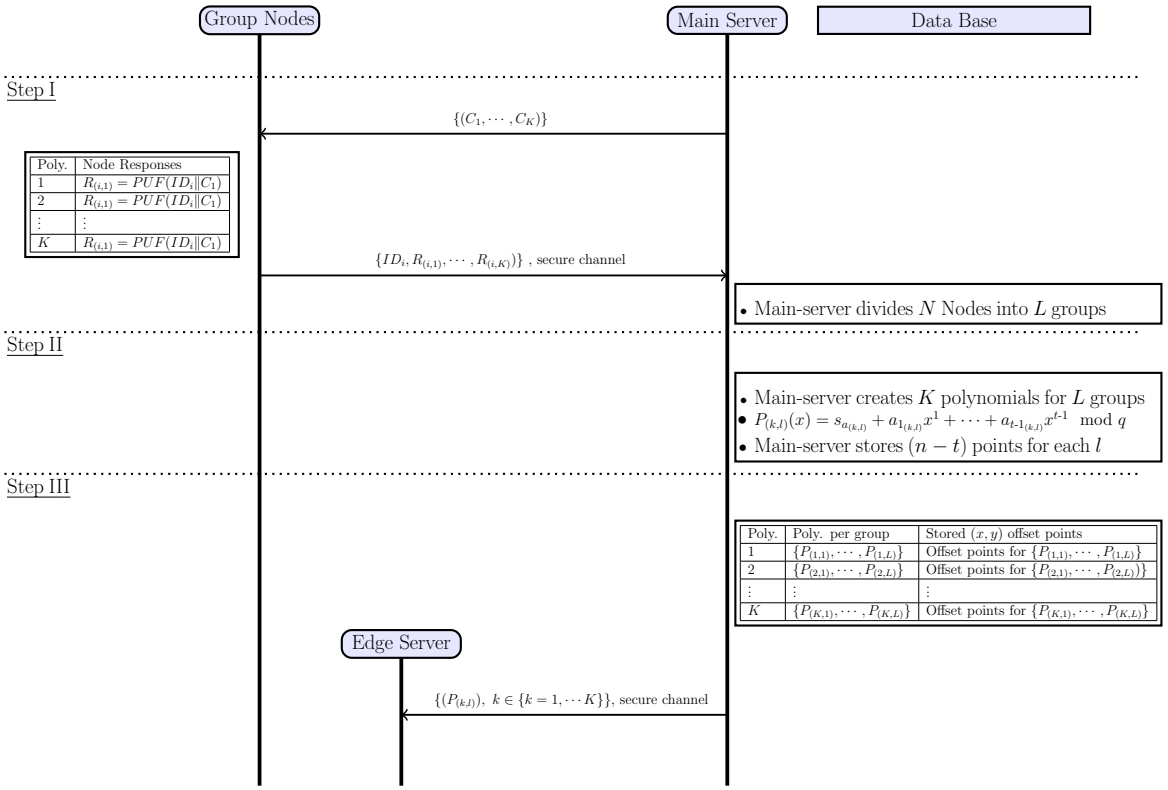

Figure 5.4: Phase I: Registration & Initialization Phase

- **Step III:** At the end of the initialization phase, the main server has a reservoir of $K \times L$ polynomials as shown in Table 5.4. The main-server sends the edge-server a copy via a secure channel. The initialization process is shown in Fig. 5.4.

Table 5.4: Polynomial Set Reservoir

Poly.	Poly. per group	Stored (x, y) offset points
1	$\{P_{(1,1)}, \cdots, P_{(1,L)}\}$	Offset of $\{(P_{(1,1)}, \cdots, P_{(1,L)})\}$
2	$\{P_{(2,1)}, \cdots, P_{(2,L)}\}$	Offset of $\{(P_{(2,1)}, \cdots, P_{(2,L)})\}$
$\vdots$	$\vdots$	$\vdots$
K	$\{P_{(K,1)}, \cdots, P_{(K,L)}\}$	Offset of $\{(P_{(K,1)}, \cdots, P_{(K,L)})\}$

5.5.3 Group Authentication & Key-agreement Phase

The details for the authentication process are shown in Fig. 5.5 and as follows.

- **Step I:**

  a. When the edge-server is required to authenticate a group, it sends the selected polynomial challenge C_k, an index b that represents the authentication session, and a request for nodes' IDs. Also, the edge-server generates a random group-token for this session, Tkn and sends its $H(Tkn)$.

  $ES \rightarrow Nodes: \ < C_k, b, H(Tkn), IDs\ Request >$

  Harn [71] defines a *t-Secure m-User n-Group* Authentication Scheme (t, m, n)-GAS to have the following two properties:

  (a) the scheme must resist up-to (t-1) colluding group-members, i.e. t is the threshold,

  (b) m users can determine whether these Users belong to the group with n members.

  Furthermore, Harn states that the *correctness* of GAS scheme is positive if all Users are group-members; otherwise, Users are not group-members.

  b. Each m participating node sends its ID back to the edge-server.

  $Node_i \rightarrow ES: \ < ID_i >$

  c. The edge-server checks if all IDs belong to the same group. If yes, the edge-server continues; otherwise, it rejects the foreign IDs. Then, the edge-server checks if the number of participating nodes in the authentication process is greater than the polynomial threshold, i.e. $m \overset{?}{\geq} t$. If the check passes, the edge-server moves to the second step; otherwise, it requests more nodes to join the authentication process. The edge-server halts the authentication process

until it has at least t participating node.

- **Step II:**

  a. The edge-server sends each participating node its hashed response xor-ed with this token. The edge-server also sends the nodes that have offsets their values xor-ed with the group-token.

  $$ES \rightarrow Nodes\colon\ [ID_i, OT_i, hd_i, (ID_j, O_j, H(\text{Offset}_j))]$$

  where $OT_i = Tkn \oplus H(R_{(i,k)})$, hd_i is the helper-data vector, $O_j = Tkn \oplus$ *offset-to-poly*, and j is the index for nodes with polynomial offsets.

  b. Each participating node first recovers the group-token, $Tkn^{'} = OT_i \oplus H(R_{(i,k)})$. Then, each node checks the integrity of Tkn by $H(Tkn^{'}) \stackrel{?}{=} H(Tkn)$. Also, for nodes with $\text{Offset}_j^{'} = O_j \oplus Tkn$, the nodes check the integrity of *Offset* by $H(\text{Offset}_j^{'}) \stackrel{?}{=} H(\text{Offset})$. Consequently, each node is able to compute its $P_{(k,l)}(x)$ share by applying Equation (5.1). All participating nodes can compute the group-token, and the nodes which need the offset received from the edge-server add the offsets to their PUF-responses, respectively, as shown in Equation (5.3).

  $$R_{(k,i,j)} \quad = PUF(H(ID_i \| C_k)) + Offset_j \tag{5.3}$$

  c. The construction of the secondary polynomial is similar to the construction of the first polynomial $P_{(k,l)}(x)$, except that the y-points of the m-participating nodes are derived from the hashes of the nodes original CRPs. The coefficients $\{c_{1(k,l)}, \ldots, c_{m-1(k,l)}\}$ are also computed linearly by $\mathbf{c} = (\mathbf{x^T}.\mathbf{x})^{-1}\mathbf{x^T}.\mathbf{y_b}$ where $\mathbf{x}$

is the Vandermonde matrix of x-points, $\mathbf{y_b}$ is the vector of hashed-y-points, and $\mathbf{c}$ is the vector the polynomial's coefficients. The format is shown in Equation (5.4).

$$P_{(k,l,b)}(x) = \begin{array}{l} s_{b_{(k,l)}} + c_{1_{(k,l)}} x^1 \cdots + \\ c_{m\text{-}1_{(k,l)}} x^{m\text{-}1} \mod q \end{array} \tag{5.4}$$

d. Similarly, each participating node computes its two shares of the two polynomials, $P_{(k,l)}$ and $P_{(k,l,b)}$, according to Equations (5.5) and (5.6), respectively.

$$y_{(i,k,l)} = R_{(i,k,l)} = PUF(H(ID_i \| C_k)) + O_j \tag{5.5}$$

$$y_{(i,k,l,b)} = H^b(R_{(i,k,l)}) = H^b(PUF(H(ID_i \| C_k))) \tag{5.6}$$

e. The edge-server computes the sum of the two polynomials secrets as shown in Equation (5.7) corresponding to group l.

$$Sum_{(k,l)} = s_{a_{(k,l)}} + s_{b_{(k,l)}} \tag{5.7}$$

f. The edge-server sends the group-nodes the following broadcast which includes the session nonce n_a, and $H(Sum \| n_a)$, respectively.

$ES \rightarrow Nodes$: $[n_a, S{=}H(Sum_{(k,l)} \| n_a)]$

g. Each group-node broadcasts the sum of its two shares xor-ed with the hash of the current token as follows.

$Node_i \rightarrow ES, Group$:
$\{C_i, (y_{(i,k,l)} + y_{(i,k,l,b)}) \oplus H^b(Tkn)\}$

- **Step III:**

  a. Each group-node first retrieves other nodes sum-of-shares by xor-ing $H^{(b)}(Tkn)$ from their broadcasts, applies Lagrange's interpolation polynomial to compute the $Sum_{(k,l)}$, and checks if the sum matches the ES broadcast, see Equation (5.8). If the check passes, then the many-to-many group authentication is successful; otherwise, the edge-server communicates with the main-server of a failed authentication.

$$\begin{aligned} &Sum^{'}_{(k,l)} = \sum_{i=1}^{m}(D_i) \prod_{j=1,j\neq i}^{m} \frac{x_j}{x_i - x_j} \mod q \\ &where\ D_i = (y_{(i,k,l)} + y_{(i,k,l,b)}) \\ &H(Sum^{'}_{(k,l)} \| n_a) \stackrel{?}{=} S \end{aligned} \tag{5.8}$$

  b. Both the edge-server and group-nodes compute their session keys as shown below.

$$SK_i = H(n_a \| Sum_{(k,l)} \| y_{(i,k,l,b)}) \tag{5.9}$$

  c. The group-nodes also derive the session's group key as given in Equation (5.10).

$$GK_{(k,b)} = H(n_a \| Sum_{(k,l)} \| H^b(Tkn)) \tag{5.10}$$

  d. The edge-server communicates with the group using the sessions group-key.

$$ES \rightarrow Group\colon \{Msg\}_{GK_{(k,b)}}$$

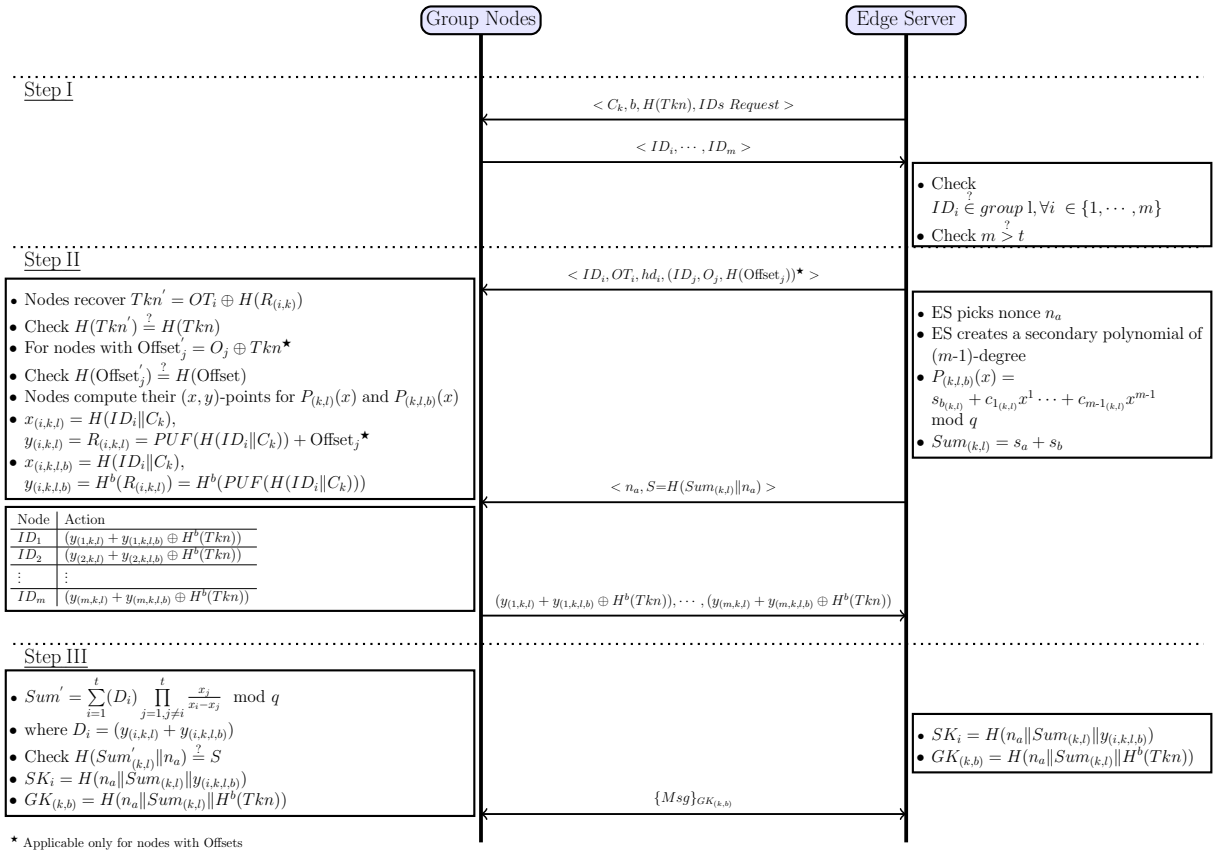

Figure 5.5: Phase II: Mutual Edge-Node Group Authentication & Key-agreement.

For each subsequent authentication with this group, the edge-server may use the same k^{th} polynomial with another b secondary polynomial, or pick another k^{th} polynomial and another secondary polynomial. After authentication, the group may transfer to another zone to get authenticated with another edge-sever.

5.5.4 Integrity Check and Session-Key Synchronization

To circumvent adversary's attempts to tamper with protocol's randomly generated values, namely, Tkn, *Offset*, and n_a, the edge-server sends the hash of these values. Specifically, at the beginning of each new session, Step I, the edge-server sends $H(Tkn)$ along with the session-challenge C_k and session-index b. Later, when each node sends its ID, the edge-server delivers to each node its $OT_i = Tkn \oplus H(R_{(i,k)})$. Upon receiving OT_i, each node computes Tkn' and compares its hash to the received $H(Tkn)$; if matched,

then integrity check on Tkn is passed, otherwise, the session is terminated. The integrity check for the value Tkn also ensures the integrity of OT_i. Similarly, to ensure the integrity of $Offset_j$, the edge-server sends the nodes that have offsets both O_j and $H(Offset_j)$; this enables these nodes to verify the value of $Offset_j$ after computing it from O_j. Specifically, each node compares the hash of its computed $Offset_j$ with the received $H(Offset_j)$. The integrity check for the value $Offset_j$ also ensures the integrity of O_j. Finally, to verify session-key synchronization, each node checks the value of Sum and n_a by comparing the calculated $Sum^{'}$ and the received n_a with the published $H(Sum|n_a)$, Step II.

5.6 Security Analysis

5.6.1 Threat Model

Our security analysis is based on the CK-adversary model [38]. In the model, an adversary $\mathcal{A}$ is given access to oracle-machines which emulate real life attacks such as eavesdropping, impersonating nodes, leaking keys, and corrupting nodes. Let Π denote the protocol and let Γ denote two participants: the edge server $\mathcal{E}$ and a group node $\mathcal{N}_i$. The CK-threat model queries are listed in Table 5.5.

5.6.2 Security Proofs

Theorem 4 *Let polynomial, P_1, that defines the secret s_a be of $(t\text{-}1)$-degree, and let polynomial, P_2, that defines the secret s_b be of $(m\text{-}1)$-degree, for $t \leq m \leq n$. The polynomial $\mathcal{P}(x) = P_1 + P_2(x)$ is a Shamir's secret-secret polynomial that is a secure against up to $(m\text{-}1)$ colluders.*

Proof:

Because our scheme has two different polynomials with two different sizes, we prove

Table 5.5: Queries Descriptions

Query	**Description**
Init(Π)	This query initializes a new protocol instance. The initialization include setting-up the group *IDs*, PUF-responses, and generating the polynomial-set.
Execute($\mathcal{N}_i, \mathcal{E}$)	This query emulates $\mathcal{A}$'s ability to eavesdrop on any communication between any group node $\mathcal{N}_i$ and the edge-server $\mathcal{E}$. This is equivalent to a passive attack.
Send(m,Γ)	This query models $\mathcal{A}$'s ability to launch an active attack. It allows $\mathcal{A}$ to impersonate any participant Γ. Specifically, it allows $\mathcal{A}$ to impersonate $\mathcal{E}$ by sending a message m to any group node $\mathcal{N}_i$. It also, allows $\mathcal{A}$'s to impersonate any group node $\mathcal{N}_i$ by sending a message m to $\mathcal{E}$. The impersonated participant responds with $m^{'}$ according to the real protocol description.
Corrupt($\mathcal{N}_i$)	When $\mathcal{A}$ uses this query, the oracle returns the state information for a node $\mathcal{N}_i$ of the current session. This include revealing the node's token and sum-of-two shares.
SKReveal($\mathcal{N}_i$)	This emulates $\mathcal{A}$'s ability to leak node $\mathcal{N}_i$ session key.
Test($\mathcal{N}_i$)	$\mathcal{A}$ can use this query only once on a fresh session. Fresh implies that *SKReveal*($\mathcal{N}_i$) has not been queried in this session. In this query, the *Test* oracle-machine flips a coin to determine the value of $B \in \{0, 1\}$. If $B = 1$, it returns $\mathcal{N}_j$ SK session key; otherwise, it returns a random number of the same length.

the theorem by satisfying two conditions [28]. 1) The knowledge of m or more sum-of-shares, recovers the sum-of-secrets, $s_a + s_b$. 2) The collusion of up to (t-1) participants, does not give the adversary, $\mathcal{A}$, any information about the two polynomials secrets, $s_a + s_b$.

1. Let s_d be the "super-secret", i.e., the sum of the two polynomial secrets $s_a + s_b$, and let $\{d_{i_1}, d_{i_2}, \cdots, d_{i_n}\}$ be the set of "super-shares" which defines s_d, and $\mathcal{P}(x)$ is its polynomial. Let s_a be the secret defined by the set of $\{a_{i_1}, a_{i_2}, \cdots, a_{i_n}\}$ shares for polynomial P_1 described in Equation (5.2), and let s_b be the secret defined by the set of $\{b_{i_1}, b_{i_2}, \cdots, b_{i_n}\}$ shares for polynomial P_2 described in Equation (5.4). Table 5.6 shows the secrets, s_a, s_b, and s_d, and their corresponding sets and polynomials.

   Let S, T be the sets of possible secrets and legitimate shares, respectively. Let the induction function be, $F_I : T \rightarrow S$, of the (t, n) SSS, such that $s = F_I(s_{i_1}, s_{i_2}, s_{i_3}, \cdots, s_{i_t})$

Table 5.6: Polynomials used in *PUF-GAS*

Polynomial	Degree	Secret	Set of all shares
$\mathcal{P}$	(m-1)	s_d	$\{d_{i_1}, d_{i_2}, \cdots, d_{i_n}\}$
P_1	(t-1)	s_a	$\{a_{i_1}, a_{i_2}, \cdots, a_{i_n}\}$
P_2	(m-1)	s_b	$\{b_{i_1}, b_{i_2}, \cdots, b_{i_n}\}$

where I is any subset of T, $I \subseteq \{1, 2, \cdots, n\}$ with $\|I\| = t$. We select the induction function to be the Lagrange's interpolating polynomial, $P(0) = \sum_{i=1}^{t} y_i \prod_{j=1, j\neq i}^{t} \frac{x_j}{x_i - x_j} \mod q$, where t is the threshold. To recover the super-secret s_d, we require m-shares, because $\mathcal{P}$ is at most (m-1)-degree polynomial. We apply the formula directly to show that the Lagrange's interpolation polynomial of the sum-of-shares equals to the sum of secrets.

$$\begin{aligned} s_d &= s_a + s_b \\ &= P_1(0) + P_2(0) \\ &= \sum_{i=1}^{m} y_{(i,k,l)} \prod_{j=1, j\neq i}^{m} \frac{x_j}{x_i - x_j} + \sum_{i=1}^{m} y_{(i,k,l,b)} \prod_{j=1, j\neq i}^{m} \frac{x_j}{x_i - x_j} \\ &= \sum_{i=1}^{m} (y_{(i,k,l)} + y_{(i,k,l,b)}) \prod_{j=1, j\neq i}^{m} \frac{x_j}{x_i - x_j} \end{aligned}$$

2. Assume that there are up to (t-1) conspirators collude to gain information on s_a or s_b. Without loss of generality, we assume that these conspirators are the first (t-1) shareholders, say $\{a_{i_1}, a_{i_2}, \cdots, a_{i_{(t\text{-}1)}}\}$ and $\{b_{i_1}, b_{i_2}, \cdots, b_{i_{(t\text{-}1)}}\}$. By the definition of (t, n) threshold SSS [160], the adversary, $\mathcal{A}$, cannot leak any information on s_a or s_b even with the knowledge of (t-1) "sub-shares". However, $\mathcal{A}$ can determine $\{d_{i_1}, d_{i_2}, \cdots, d_{i_{(t\text{-}1)}}\}$. Let us assume further that $\mathcal{A}$ is an inside-attacker who knows s_d. Thus, $\mathcal{A}$ is able to determine $\{d_{i_t}, d_{i_{(t+1)}}, \cdots, d_{i_n}\}$. However, even with this knowledge, $\mathcal{A}$'s probability of leaking s_a or s_b is negligible.

It follows that the $\mathcal{P}$ polynomial used in *PUF-GAS* is at most (m-1)-degree polynomial, and the knowledge of m "super-shares" recovers s_d. Also, the two polynomials' secrets s_a and s_b and the "sub-shares" of the two polynomials are not revealed to the adversary. ■

We define Semantic-security, Entropy-Smoothing, and Decisional-Uniqueness-Problem (DUP) Assumption, respectively, as follows [166, 203].

Definition 11 (Semantic-security) *Let Win be a winning-event in which $\mathcal{A}$ executes Test($\mathcal{N}_i$) on a fresh protocol instance and outputs B'. If $B' = B$, $\mathcal{A}$ is able to break the semantic-security of Π. We define the advantage of $\mathcal{A}$ in breaking the semantic-security of PUF-GAS as*

$$Adv_{\mathcal{A}}^{PUF\text{-}GAS} = |Pr[Win_0] - \frac{1}{2}| = |Pr[B' = B] - \frac{1}{2}|$$

Definition 12 (Entropy Smoothing) *Let $\mathcal{H}$ be a family of hash-functions. It is hard to distinguish between $(k, H_k(\delta))$ and (k, h) where k is a random number of K, δ is a random element of a finite-cyclic-group G, and h is a random element $\{0,1\}^l$. Formally,*

$$|Pr[k \xleftarrow{R} K, \delta \leftarrow G : \mathcal{A}(k, H_k(\delta)) = 1] -$$

$$Pr[k \xleftarrow{R} K, h \leftarrow \{0,1\}^l : \mathcal{A}(k, h) = 1]| \leq \epsilon_{es}$$

where ϵ_{es} is a negligible function.

Definition 13 (Decisional-Uniqueness-Problem (DUP) Assumption) *Given a challenge C, an arbitrary PUF-instance PUF_{Adv} which outputs an l-bit response, and a random number z which is uniformly distributed from $z \in \{0,1\}^l$, it is infeasible to distinguish between the tuples $(C, PUF_{Adv}, z \leftarrow \{0,1\}^l)$ and $(C, PUF_{Adv}, z = PUF_t(C))$ where PUF_t is the targeted PUF-instance to forge. Formally,*

$$|Pr[(C, PUF_{Adv}, z \leftarrow \{0,1\}^l) = 1] -$$

$$Pr[(C, PUF_{Adv}, z = PUF_t(C)) = 1]| \leq \epsilon_{puf}$$

where ϵ_{puf} is a negligible function.

Lemma 1 (Difference Lemma) *Let A, B, and F be events defined in a some probability-distribution. If A and B are identical without the occurrence of a failure-event F, i.e. $A \wedge \neg F \iff B \wedge \neg F$, then $\mid Pr[A] - Pr[B] \leq Pr[F] \mid$.*

Theorem 5 *The advantage of breaking the semantic-security of the PUF-GAS session-key for a probabilistic-polynomial-time adversary $\mathcal{A}$ is*

$$Adv_{\mathcal{A}}^{PUF\text{-}GAS} \leq 12\epsilon_{es} + 2\epsilon_{puf} + \frac{q_H^2 + (q_{snd} + q_{exe})^2 + q_{snd}}{2^l}$$

where q_H, q_{snd}, and q_{exe} are the number of Hash, Send, and Execute queries, respectively. l is the length of hash-output, PUF-response, and session-key.

Proof: We conduct our proof by the method of sequence-of-games, also referred to as game-hopping [166, 203]; the games are $Game_i$, $i \in \{0, 1, 2, 3, 4\}$. Let Win_i, $i \in \{0, 1, 2, 3, 4\}$ be the event of $\mathcal{A}$'s guessing the bit B correctly in the *Test* oracle for $Game_i$, and let $Pr[Win_i]$ represent the probability of winning the game. The details are as follows.

- ***Game*$_0$:** This game represents the real-attack performed by $\mathcal{A}$ against *PUF-GAS*. By definition, $\mathcal{A}$'s advantage in breaking the semantic-security of *PUF-GAS* protocol is,

$$Adv_{\mathcal{A}}^{PUF\text{-}GAS} = |Pr[Win_0] - \tfrac{1}{2}| \tag{5.11}$$

- ***Game*$_1$:** In this game, the Hash-queries and PUF-queries are simulated from the random-oracle-model. In other words, the PUF-responses and the hash-function outputs are generated randomly and uniformly from $\{0, 1\}^l$.
  - *Hash-oracle:* The hash-function is modeled as a truly-random function obtained from a uniformly distributed sequence from $\{0, 1\}^l$. The random-oracle can be viewed as a black-box in which a list of $(query, answer)$ $\mathcal{L}_H$ is main-

tained. When this black-box is queried with an input, the list $\mathcal{L}_H$ is checked to see if the input already exists. If yes, the same previous output is given; otherwise, a new random-sequence uniformly generated from $\{0,1\}^l$ is given, and the new $(query, answer)$-pair is augmented to the $\mathcal{L}_H$ list.

– *PUF-oracle:* Similar to the Hash-oracle, a $\mathcal{L}_{puf}$ list is also maintained by the PUF-oracle. The exception is that adversary $\mathcal{A}$ does not have a direct access to the PUF-oracle. This is because $\mathcal{A}$ is not able to leak the PUF-response even if the PUF-instance is captured, see Definition (10). Additionally, we assume that the PUF-instances are ideal and stable and thus Fuzzy-Extractors are not needed.

We note that the transition from $Game_0$ to $Game_1$ is only by the use of the Hash-oracles and PUF-oracles instead of hash-function and PUF-instances, respectively. Thus, the two games are indistinguishable except for the Entropy-Smoothing of Hash function and DUP Assumption given in Definition (12) and (13), respectively. Because *PUF-GAS* protocol has 12 Hash-queries and 2 PUF-queries, we have

$$|Pr[Win_1] - Pr[Win_0]| \leq 12\epsilon_{es} + 2\epsilon_{puf} \tag{5.12}$$

- ***Game*$_2$:** In this game, the assumptions of $Game_1$, i.e. using Hash-oracles and PUF-oracles, are maintained but with the exclusion of collisions possibilities. According to the birthday-paradox, if at most q queries are requested from a random-oracle of l-bits, then the probability of collision is $\frac{q^2}{2^{l+1}}$. The collision may occur in our transcript if two honest instances select the same random number. Specifically, if $\mathcal{E}$ picks a previously used Tkn in $MSG1 : [ID_i, OT_i, hd_i, (ID_j, O_j, H(\text{Offset}_j))]$ there may be a transcript collision. Similarly, if $\mathcal{E}$ picks previously used n_a in $MSG2$: $(n_a, H(Sum_{(k,l)} \| n_a))$, there may be a transcript collision. Given that there could be at most $(q_{snd} + q_{exe})$-pairs of random numbers generated in the transcript,

the probability of collision is $\frac{(q_{snd}+q_{exe})^2}{2^{l+1}}$. Also, the output of the Hash-queries could cause collision. Using the Difference-Lemma (1), we get,

$$|Pr[Win_2] - Pr[Win_1]| \leq \frac{q_H^2}{2^{l+1}} + \frac{(q_{snd}+q_{exe})^2}{2^{l+1}} \tag{5.13}$$

- ***Game*$_3$:** In this game, we keep the assumptions of $Game_2$, i.e. all random numbers used in the sessions are fresh and have not been used before. However, in this game, we exclude the probability of $\mathcal{A}$ getting lucky in guessing the session-key. The probability for this to occur is $\frac{q_{snd}}{2^l}$. Thus, using the Difference-Lemma (1), we get

$$|Pr[Win_3] - Pr[Win_2]| \leq \frac{q_{snd}}{2^l} \tag{5.14}$$

- ***Game*$_4$:** In this game, we are able to finally tie the semantic-security of the *PUF-GAS* protocol with the hardness of the Hash-function. Specifically, the derivation of the session-key, $SK = H(n_a \| Sum_{(k,l)} \| y_i)$ requires the knowledge of three random values, n_a, $Sum_{(k,l)}$, and y_i. In $Game_1$, we showed that the PUF-responses are simulated based on the random-oracle-model; thus, y_i is random-sequence of bits of length l. In $Game_2$, we excluded the possibilities of collisions on $Sum_{(k,l)}$ or n_a. Thus, all randomly generated nonces are fresh in each session. Also, in $Game_2$, we showed that all Hash-queries are generated using the random-oracle-model. Thus, when $\mathcal{A}$ is challenged in the *Test($\mathcal{N}_i$)* oracle, $\mathcal{A}$ cannot distinguish between the protocol's SK from a random-sequence of bits of the same length, because from the point view of $\mathcal{A}$, they are both random-numbers. Thus,

$$|Pr[Win_4]| = \frac{1}{2} \tag{5.15}$$

Using the triangular-inequality reduction on Equations (5.11) to (5.15), we get:

$$Adv_{\mathcal{A}}^{PUF\text{-}GAS} = |Pr[Win_0] - \frac{1}{2}|$$

$$= |Pr[Win_0] - Pr[Win_4]|$$

$$\leq 12\epsilon_{es} + 2\epsilon_{puf} + \frac{q_H^2 + (q_{snd} + q_{exe})^2 + q_{snd}}{2^l}$$

■

5.6.3 Analysis with AVISPA

AVISPA is an automated software widely used for security protocol verification [14, 18, 170]. The tool searches for potential attacks such as Man-in-the-Middle attack, replay attacks, active and passive attacks.

The details of our *PUF-GAS* AVISPA implementation are as follows.

1. We implemented a group of four IoT-nodes, an edge-server, and a main-server, namely A, B, C, D, ES, and MS, respectively.

2. We abstracted the construction of the PUF-device and considered the node's PUF-response as an internal secret exchanged between the node and the main-server. Specifically, the PUF-responses associated with the four nodes, P_a, P_b, P_c, and P_d are locally generated pseudo-random numbers in response to the main-server challenge, C_k.

3. We also considered the group-token as a secret initially delivered from the main-server to the four IoT-nodes and the edge-server.

4. It is proven that reconstructing a (t-1)-degree polynomial requires the knowledge of t points [30, 34, 160]. Consequently, since AVISPA has several limitations including the lack of algebraic operation support, we model such a polynomial reconstruction by a variable X whose value is equal to the xor of t values $\{x_1, x_2, \ldots, x_t\}$, i.e.,

$X = x_1 \oplus x_2 \oplus \cdots \oplus x_t$, because similar to recovering (t-1)-degree polynomial, recovering X requires the knowledge of all the t x_i values.

5. The recovered secret is used to derive the group-session-key, GK. Msg is encrypted using the derived group-session-key, and delivered from ES to node A.

6. The security goals in the HLPSL code are the secrecy and authenticity of $\{P_a, P_b, P_c, P_d \; and \; Msg\}$. Specifically, P_a, P_b, P_c are secrets delivered from IoT-nodes to the main-server, and Msg is a message delivered with the derived group-session-key. Note that since the message Msg is encrypted with the derived group-session-key GK, therefore, the secrecy of Msg implies the secrecy of the group-session-key.

We ran the protocol under CL-AtSe and OFMC AVISPA back-end simulator, and with successful results. Please note that other back-end AVISPA simulators, SATMC and TA4SP, do not support xor operation, and thus we cannot run our protocol under these two back-ends [170]. The result of the AVISPA CL-AtSe back-end analyzer simulation is shown in Table 5.7 and for AVISPA OFMC is in Table 5.8.

Modeling Attack: *PUF-GAS* resists PUF modeling attacks. This is because our *PUF-GAS* protocol does not reveal any of the node's CRPs, instead, only the node's sum-of-two-shares xor-ed with the session token is revealed. Because the revealed value is a random value sampled uniformly from $\{0,1\}^\lambda$, the protocol is resistant to modeling attacks.

5.7 Experimental Results and Comparative Evaluation

In this section, we present our experimental results and show a comparative evaluation of our protocol in terms of security features, computational cost, communication overhead, and storage.

Table 5.7: AVISPA CL–AtSe Simulation Results

Description	**AVISPA Simulation Results**
SUMMARY	
	SAFE
DETAILS	
	BOUNDED_NUMBER_OF_SESSIONS
	TYPED_MODEL
PROTOCOL	
	/home/span/span/testsuite/results/PUF-GAS_4_3.if
GOAL	
	As Specified
BACKEND	
	CL–AtSe
STATISTICS	
	Analysed : 67 states
	Reachable : 13 states
	Translation: 2831.55 seconds
	Computation: 2831.55 seconds

We implement the following security operations on Arduino-Mega, an 8-bit RISC-based micro-controller ATmega2560, 8KB SRAM, and 16MHz clock-speed using the Arduino Cryptography Libraries [13, 185]. Table 5.9 shows execution times of Crypto-operations and SRAM-PUF measurements based on BCH $(63, 7, 31)$ error correction code [157]. To compute the results shown in Table 5.10, and for a fair comparison with other protocols, we adopt Chien's [105] assumption of considering $T_{inv,q} \cong 240T_{mul,q}$. On the other hand, for our (128-bit Arbiter) SRAM-PUF computations, we scale Setyawan's [157] results of reconstructing an SRAM-PUF key of 256 bits to suit our 128-bits key.

The comparative evaluations are shown in Table 5.10 and Table 5.11 for security and performance, respectively. Table 5.10 shows a comparison between our scheme and other related schemes in terms of performance. In what follows, we show our comparative analysis for computation complexity, communication overhead, and storage with [21, 23, 48, 105, 117, 128] in which only [21, 105, 128] are GAS schemes, and oth-

Table 5.8: AVISPA OFMC Simulation Results

Description	**AVISPA Simulation Results**
SUMMARY	
	SAFE
DETAILS	
	BOUNDED_NUMBER_OF_SESSIONS
	TYPED_MODEL
PROTOCOL	
	/home/span/span/testsuite/results/PUF-GAS_4_3.if
GOAL	
	As Specified
BACKEND	
	OFMC
COMMENTS	
STATISTICS	
	parseTime: 0.00s
	searchTime: 28.54s
	visitNodes: 285 nodes
	depth: 25 piles

Table 5.9: Arduino-Mega ATmega2560 Simulation Results

Operation	**Symbol**	**Execution time**
Hash SHA-256[1]	T_h	$5.34624msec$
Symmetric-key encryption[2]	T_{enc}	$1.14432msec$
Symmetric-key encryption[2]	T_{dec}	$2.17056msec$
Modular multiplication[3]	$T_{mul,q}$	$1.3863msec$
Modular addition[3]	$T_{add,q}$	$93.10\mu sec$
Modular exponentiation [3]	T_{exp}	$1.470816sec$
ECC multiplication[4]	EC_{mul}	$32.308msec$
ECC addition [4]	EC_{add}	$24.047msec$
Key-reconstruction SRAM-PUF [5]	T_{Rec}	$488.97msec$
SRAM-access [6]	T_{sram}	$0.277msec$

[1]: SHA-256 over 32-bytes data.
[2]: AES-128-ECB, key-size= 128-bits.
[3]: Modulus size= 256 bits.
[4]: ECC Curve-25519.
[5]: Based on [157] Stage 3 SRAM-PUF key reconstruction using BCH=$(63, 7, 31)$=$977.97msec$ for 256-bits key.
[6]: Time accessing 256-bytes on Arduino-Mega internal memory.

.

Table 5.10: Comparative Evaluation

Type	Ref	Storage-Node	Storage-Server▼	Computation Complexity-Node▲	Computation Complexity-S▲▼
GAS	This work	*None*	n	$[1\times T_{rec}+(2b+4)\times T_h+(m)\times T_{add,q}+[2(m-1)+2]\times T_{mul,q}+T_{inv,q}]\sim 0.91104sec$	$[(2b+4)\times T_h+(m)\times T_{add,q}+[2(m-1)+2]\times T_{mul,q}+T_{inv,q}]\sim 0.422146sec$
	Aydin [21]	1 *share*	n	$[(2(m-1)+2)\times T_{mul,q}+1\times T_{inv,q}]+3\times EC_{mul}+(m-1)\times EC_{add}+1\times T_h\sim 0.9473sec$	$[(2(m-1)+2)\times T_{mul,q}+1\times T_{inv,q}]+3\times EC_{mul}+(m-1)\times EC_{add}+1\times T_h\sim 0.9473sec$
	GASE [128]	2 *shares*	n	$[(2(m-1)+2)\times T_{mul,q}+1\times T_{inv,q}]+(m-1)\times T_{add,q}+(m-2)\times T_h\sim 0.4862sec$♠	$[(2(m-1)+2)\times T_{mul,q}+1\times T_{inv,q}]+(m-1)\times T_{add,q}+(m-2)\times T_h\sim 0.4862sec$♠
	Li [105]	θ *shares*	n	$\theta[(2(m-1)+2)\times T_{mul,q}+1\times T_{inv,q}]+(m-1)\times T_{add,q}+1\times T_h\sim 1.1858sec$	$\theta[(2(m-1)+2)\times T_{mul,q}+1\times T_{inv,q}]+(m-1)\times T_{add,q}+1\times T_h\sim 1.1858sec$
Mutual Auth.	Chen [48]✦	*None*	$n\times\Psi$ ✦	$(m-1)\times[\Psi\times T_{Rec}]\sim 185.81sec$	$(m-1)\times[C_{\Psi}^{m+1}\times[(m-1)\times T_{add}+[2(m-1)+2]\times T_{mul,q}+T_{inv,q}+T_h]]\sim 157.7sec$■
	Mahalat [117]✦	*None*	$n\times\Psi$	$(m-1)\times[1\times T_{Rec}+3\times T_{exp}+4\times T_{mul,q}+2\times T_{add,q}+2\times PV]\sim 94.24sec$✪	$(m-1)\times[1\times T_{Rec}+3\times T_{exp}+4\times T_{mul,q}+2\times T_{add,q}+2\times PV]\sim 94.24sec$✪
	Bansal [23]✦	1 *key*	$n\times\Psi$	$(m-1)\times[\Psi\times T_{Rec}]\sim 185.81sec$	$(m-1)\times[(m-1)\times T_{add}+[2(m-1)+2]\times T_{mul,q}+T_{inv,q}+T_h]\sim 7.51sec$

▼: S is for edge-server, main-server, sink-hole, or group-leader of the protocol.
▲: Computations for authentication and key-agreement. Parameters are $m=20, \theta=3$, and $b=1$.
✦: For mutual-authentication-scheme, we multiply by (m-1). Ψ is the number of CRPs deployed for one node.
✪: Based on PV Pedersen's VSS $\sim$ (m-1) $\times T_{mul,q}$
■: For the combination formula of the protocol C_{Ψ}^{m+1}, we selected values form the publication [48] for m and Ψ Such that $m+1=21$ and $\Psi=20$ such that $C_{\Psi}^{m+1}=21$.
♠: *PUF-GAS* requires no share-storage at the node, in which if a node is stolen, the shares are never leaked. Also, [128] does not have group-key-agreement.

Table 5.11: Summary Comparison with related schemes

Parameters	Chen [48]	Aydin [21]	Mahalat [117]	Bansal [23]	GASE [128]	Li [105]	This work
Initial share delivery	Secure-channel	Secure-channel	Secure-channel	Secure-channel	Secure-channel	Secure-channel	Not required
Security-primitive	SSS & Shuffle	SSS & ECC	VPSS & PUF	SSS & PUF	SSS & Ag-MAC	SSS & ECC	SSS & PUF
Group Authentication	✗	✓	✗	✗	✓	✓	✓
Revoking/joining	✗	✓	✗	✗	✗	✗	✓
Key agreement	✗	✓	✗	✗	✓	✓	✓
Group-key agreement	✗	✗	✗	✗	✗	✗	✓
Shares update	✗	✗	✗	✗	✓	✗	Not required

ers [23, 48, 117] are mutual authentication schemes. Table 5.11 shows that our *PUF-GAS* scheme is the only scheme that does not require any initial secure share-delivery or share-update mechanism. Additionally, *PUF-GAS* is among the few schemes that are based on symmetric-key cryptography.

Computation complexity: We show the computation complexity for both the group-node and the edge-server in Table 5.10. For m participants to compute $Sum'_{(k,l)}$, shown in Equation (5.8) similar to Chien [51], each m node must perform $[(m-1)\times T_{add,q}+[2(m-1)+2]\times T_{mul,q}+T_{inv,q}]$ operations, where $T_{add,q}$, $T_{mul,q}$, and $T_{inv,q}$ represent, respectively, addition, multiplication, and multiplicative inverse in q, $GF(q)$. Also, each

node must compute its two polynomial shares and add them together which equals $[1 \times T_{rec} + (2b+3) \times T_h + 1 \times T_{add,q}]$ where T_h and T_{rec} represent hash and PUF call operations, respectively. Thus, the total operations for each node in our *PUF-GAS* is $[1 \times T_{rec} + (2b+4) \times T_h + (m) \times T_{add,q} + [2(m-1)+2] \times T_{mul,q} + T_{inv,q}]$. The computation complexity for the edge-server is $[(2b+4) \times T_h + (m) \times T_{add,q} + [2(m-1)+2] \times T_{mul,q} + T_{inv,q}]$.

On the other hand, in Aydin *et al.* key-agreement, each node computes the same basic operations for the computing the Lagrange's interpolation polynomial as well as $3 \times EC_{mul}$ and $(m-1) \times EC_{add}$, but without the PUF operation. Similarly, Li's scheme also requires for each node to store θ-polynomials secrets to compute θ- Lagrange's interpolation formula. Thus, our scheme has less computations than the latter. *GASE* [126] has $(m+2)$ hashes which increases the execution time.

Communication overhead: We compute the communication overhead for the steps shown in Section 5.5.3, I-III as follows. We assume that the $n_a = 32$ *bytes*, *ID*= 3 *bytes*, *SHA*-256 = 32 *bytes*, hd_i is $(37 \times 63 = 292$ *bytes*$)$, and all other parameters such as C_k, b, OT_i, O_j are 32 *bytes* each. Thus, the total communication overhead is computed as $[2 \times 32 + (m \times 3) + (m \times 3) + (m \times 32) + (m \times 292) + \phi \times (3 + 2 \times 32) + (3 \times 32) = (160 + 330m + 227\phi)$ *bytes*. where ϕ is the number of offset points. On the other hand, both Aydin and Li's schemes has communication overhead of (m-1); however, both of their schemes are valid for one-time group-authentication in which any subsequent authentication requires refreshing the secret, while *PUF-GAS* scheme is valid for multiple-time group-authentications.

Storage requirements: Our scheme does not require any share storage at the node; instead, each node derives its share from its PUF-response. The schemes in [21, 105, 128] require to store at least one-share. However, the edge-server in our scheme is required to store K secret polynomials for each group and the offset points; this is equivalent to storing n points for each polynomial.

5.7.1 Discussion

To the best of our knowledge, our *PUF-GAS* scheme is the only protocol that utilizes Shamir's SS and PUFs for group-authentication. To have a fair comparison between our scheme and others, we chose from the literature, only protocols that are designed for either group-authentication using SSS or mutual-authentication using PUFs and SSS. The comparison shown in Table 5.10 is based on our experimental results of running cryptography security operations on a low-end 8-bit RISC-based micro-controller ATmega2560, 8KB SRAM, and 16MHz clock-speed. The comparative result shows that our *PUF-GAS* out-performs Aydin's, Li's, Mahalat's schemes, [21, 105, 117], respectively, for both IoT-group nodes and edge-server. On the other hand, *PUF-GAS* does not out-perform *GASE*'s performance at the node; however, *PUF-GAS* does not require any share-storage at the node while *GASE* protocol requires two-shares storage for the same polynomial. Indeed, this is an advantageous security issue for *PUF-GAS*, because if a node is stolen, the shares are never leaked. Additionally, *GASE* does not support a group-key agreement or a flexible node evicting mechanism while *PUF-GAS* derives a group-key and has a simple node-evicting/joining the group due to the PUF and SSS-homomorphism. Furthermore, *PUF-GAS* supports simple multiple-time group-authentication with the same set of shares by utilizing PUFs and SSS-homorophism. On the other hand, both Chen's and Bansal's schemes [23, 48] are mutual authentication schemes and not group-authentication schemes.

5.8 Summary

We presented a construction of Shamir's secret-sharing scheme using PUFs to avoid storing secret-shares. We used this construction to design a lightweight group-authentication-scheme utilizing SSS $(+,+)$-homomorphism property. Our *PUF-GAS* scheme is suitable for three-tier cloud-edge-IoT authentication applications. We reported AVISPA

analysis to prove that *PUF-GAS* achieves both message security and authenticity. We also proved that our scheme is secure against group-key leakage and impersonation attack. We presented a comparative evaluation of our scheme with other schemes proposed in the literature, in which we showed that our *PUF-GAS* out-performs other schemes in terms of execution time. We showed our scheme is efficient and does not require share-storage or share re-distributions for any group node.

Chapter 6

Lightweight Pseudo-random-function-based Authentication and Key Agreement Protocol

6.1 Introduction

In this chapter, we propose a symmetric key mutual edge-IoT authentication and key agreement protocol based on pseudo-random function which provides anonymity with respect to any external adversary, forward secrecy, and backward secrecy. We prove the security properties of our protocol and present an evaluation of Pseudo-Random-Function Symmetric-based authentication and Key Agreement *PRF-SAKE* in terms of run-time, communication overhead, and memory space requirements, and compare it to other lightweight protocols.

6.2 Related Work

Most of the current security solutions are not suitable for edge computing paradigms for two reasons. First, most of the solutions are based on Public Key Cryptography (PKC) such as ECC (Elliptic Curve Encryption), DHKE (Difffie-Hellmen key exchange) and IBE (Identity Based encryption) (e.g., see [43, 45, 78, 80, 184]), and these solutions are not suitable for low-end limited resources devices. Second, even with symmetric key cryptography protocols, some imperative security properties such as forward and backward secrecy are not satisfied. For example, Ibrahim [76] proposed a symmetric-key based protocol in which fog-users are required to hold a long term key with the fog-server where the session keys are delivered and encrypted with this long-term key. However, the leakage of this long-term key compromises past and future messages. Wang *et al.* [179] proposed a mutual authentication edge computational offloading protocol which uses a hash chain to track the charges and the number of usage; nevertheless, forward and backward secrecy are not applicable in this protocol. Additionally, their protocol does not establish a session key. Other lightweight protocols such as [9, 67], which use Physical Unclonable Functions, achieve mutual authentication but not backward secrecy. In addition, these solutions rely on the installation of PUF units inside all IoT devices which may require special hardware manufacturing processes. Also, PUFs are sensitive to environment factors and noise which require the use of additional security primitives such as fuzzy extractors [63, 66]. In [126], the authors propose an edge computing group key agreement protocol; however, it is designed for broadcast communication and does not address backward secrecy. Similarly, the protocol in [156] is designed for roaming computation offloading services.

In this work, we propose a lightweight symmetric key authentication and key agreement protocol based on pseudo random functions which provides anonymity, backward secrecy, and forward secrecy.

6.3 System Model, and Design Goals

In our model, each edge has several static IoT devices associated with it. The assignment of IoT devices to edge entities is based on geographical proximity during the registration and initialization phase. Furthermore, the IoT nodes in the network are considered to be low-end devices with small capabilities such as small sensors, medical implanted devices, and video streaming CCTV. We further assume the existence of a trusted authority (TA) which assigns IoT nodes to its proper edge and initializes the protocol parameters. Additionally, the IoT devices or edge entities communications with the TA occur only in the initialization and registration phase. Thereafter, all edge-IoT communications are independent of the TA. Nevertheless, the TA can trace the communication and recover session keys for any disputable situations. In this chapter, we focus on mutual authentication and key agreement between IoT devices and their edge owner to offload excess data/computation. Because of the nature of target applications of our edge model, the communication between the IoT devices and their edge owner needs to be private and anonymous. Our design and security goals are summarized as follows: (i) Mutual authentication and integrity; in our protocol, we require IoT-edge mutual authentication in order to establish secure decentralized communication. (ii) Confidentiality; in the three-tier cloud-edge-IoT network, the ubiquitous things can carry confidential information such as identities, medical insurance number, bank account, and location. (iii) Backward and forward Secrecy; forward secrecy specifies that the leakage of current session or long-term secrets does not affect the old-encrypted messages. On the other hand, if the leakage of current session key or long-term secrets does not reveal future encrypted messages, this achieves backward secrecy [19, 37]. Even though both properties are imperative to privacy, only a few schemes consider backward secrecy. In our protocol, we consider both forward and backward secrecy. (iv) Anonymity and conditional traceability, these security objectives are important in static network model in which the adversary can easily trace repeated communications between an IoT device and its edge owner if

real identities are disclosed. Thus, we implement a fresh pseudo-identity for each IoT node in each new session. (v) Efficiency, because IoT devices have limited resources, our main objective is to achieve the aforementioned security goals using simple security primitives. Our lightweight protocol is based on symmetric key encryption such that its execution consumes low energy, reduces communication bandwidth, and occupies small memory space.

6.4 Proposed Scheme

6.4.1 System Overview

There are three major entities in our network, central trusted authority (TA) associated with the cloud server, authenticating edge, and several ubiquitous IoT devices. To achieve mutual authentication between the edge and the IoT device, we have one PRF-chain shared between the edge and the IoT node as shown in Fig. 6.1, where $AK_{i,t}$ is the authentication key for node i at time t, $DK_{j,t}$ is the delivery key for edge j at time t, and IV_i is the initial chain value for node i. We run our protocol in three phases, the registration phase, the authentication and key agreement phase, and the data offloading phase.

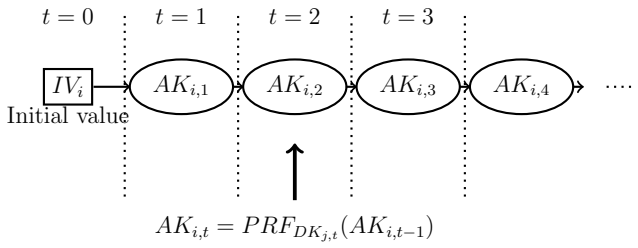

Figure 6.1: Authentication PRF-chain

The keys in our protocol are described below.

1. Long-term key/Seed: The long-term key in the network is the seed which is pre-loaded in all network entities and nodes. Nevertheless, the leakage of this seed does

not breach security properties as shown in later sections.

2. Temporary keys: The temporary keys in our protocol evolve with each new session [19].

   - Delivery Key (DK): The initial delivery key, DK_0, is generated by the TA and delivered to both the edge and IoT device via a secure channel. However, all subsequent delivery keys are generated by the edge and encrypted with the previous delivery key.
   - Authentication Key (AK): The initial value, IV, is generated by the TA and delivered to the edge and the IoT device via a secure initializing channel. However, in subsequent sessions, the edge generates this key and delivers it to the IoT node to compute the current authentication key as follows.

$$AK_{i,t} = PRF_{DK_{j,t}}(AK_{i,(t-1)}), \tag{6.1}$$

     where i indicates the node index, j indicates edge index, and t is the time index.

   - Session Key (SK): The session key is derived from random values newly generated in each session from the edge and the IoT device.

$$SK_{i,t} = eR_{j,t} \oplus nR_{i,t} \tag{6.2}$$

     where i indicates the node index, j indicates the edge index, and t is the time index. $eR_{j,t}$ and $nR_{i,t}$ are random numbers generated by the edge and the IoT device, respectively.

   - IoT Pseudo-identity: On each new session, the IoT device presents a new

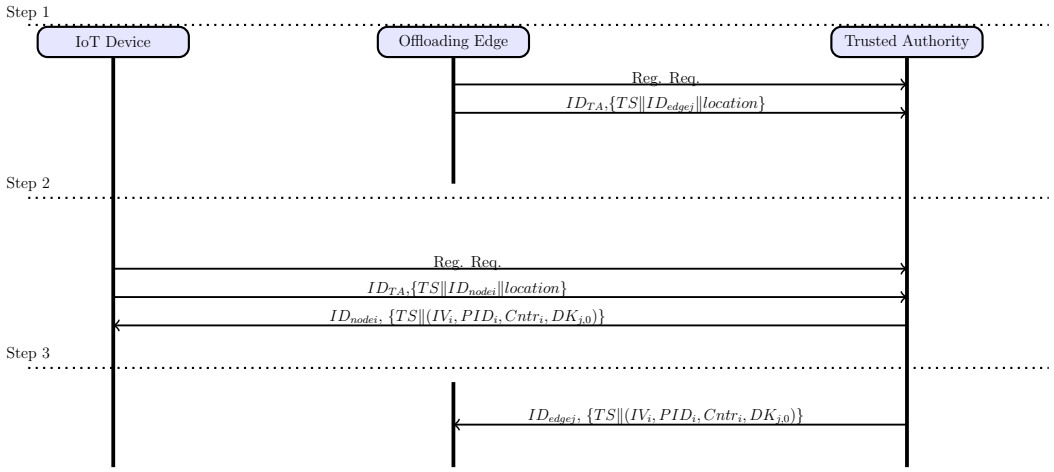

Figure 6.2: Phase I: Registration & Initialization Phase - Secure Channel

pseudo-identity to the edge based on a counter established between them.

$$PID_{i,t} = h(Cntr_{i,t} \| PID_{i,(t-1)}) \tag{6.3}$$

where $h(.)$ is a one-way hash function and $Cntr$ is the counter set in the initialization phase between the edge and the IoT node.

6.4.2 Registration & Initialization Phase

In this phase, both the edge and the IoT node register with the TA which assigns each node its edge owner based on its geographical proximity. Additionally, the TA generates the initial values such as the initial counter, $Cntr_{i,0}$, initial delivery key, $DK_{j,0}$, and starter chain value, IV_i, and delivers them in a secure channel. The details of each step are shown in Fig. 6.2 where TS stands for the time stamp.

6.4.3 Authentication and Key Agreement Phase

After the registration phase, both edge and IoT node have the initial copies of the delivery key, the pseudo-identity, and the counter. In the authentication and key agreement phase, edge and IoT node mutually authenticate each other as well as establish

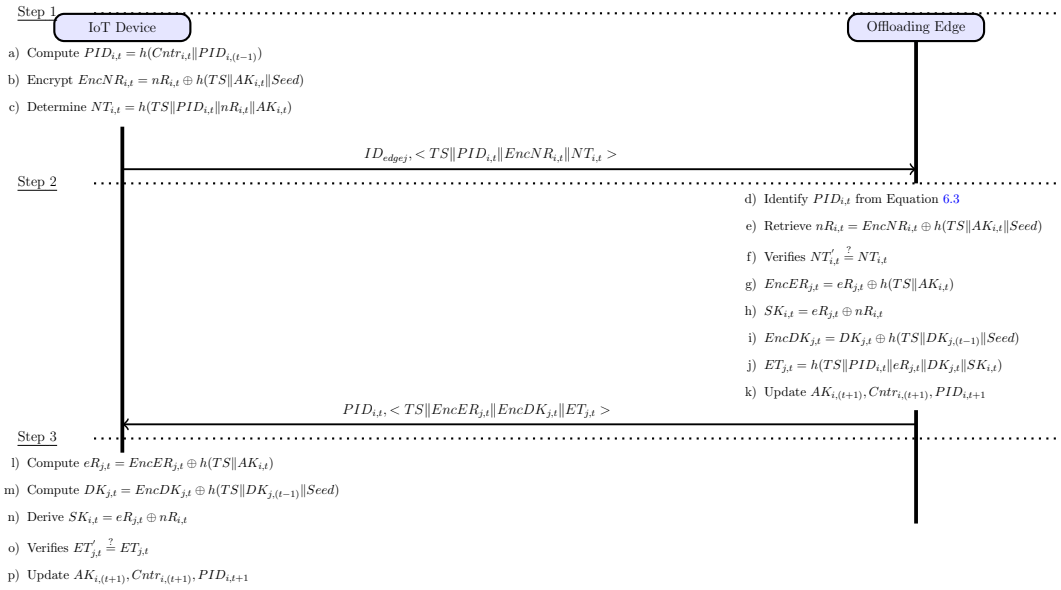

Figure 6.3: Phases II: Key Agreement & Authentication Phase

a session key for their future communications. The steps are described in Fig. 6.3.

6.4.4 Authenticated communication Phase

After authentication and session key agreement, both IoT devices and their edge owners communicate without going back to the trusted authority or the cloud.

6.5 Security Analysis

In what follows, we analyze the security properties of our *PRF-SAKE* protocol; specifically, we prove that our protocol provides mutual authentication between edge and IoT-node, forward secrecy, backward message secrecy, and anonymity with respect to any external adversary.

Definition 14 (One-way function) *A function* $f : \{0,1\}^{\lambda} \times \{0,1\}^{*} \rightarrow \{0,1\}^{\gamma}$ *is a one-way function if, for all probabilistic-polynomial-time adversary,* $\mathcal{A}$*, the advantage* $Adv_f(\mathcal{A}) = Pr[x' \leftarrow \mathcal{A}(f(x)) : x' \in f^{-1}(f(x))] \leq \epsilon(\lambda)$ *where* $\epsilon(\lambda)$ *is a negligible function.*

Definition 15 (Mutual Authenticity of *PRF-SAKE*) *We say that the PRF-SAKE protocol provides mutual authentication between an IoT-node i and an edge j if at any given time t and for any probabilistic-polynomial-time adversary $\mathcal{A}$ that can inject messages in the network, $\mathcal{A}$ cannot impersonate any IoT-node or edge , formally,*

$$Pr[\{\mathcal{E}^{\mathcal{A}}_{i/j,t}, \mathcal{T}^{\mathcal{A}}_{i/j,t}\} \leftarrow \mathcal{A}(1^{\lambda}, DK_{j,t}, Seed) \colon (\mathcal{T}^{\mathcal{A}}_{i/j,t} = \mathcal{T}_{i/j,t})] \leq \epsilon(\lambda)$$

where i is the IoT-node index, j is the edge index, $\mathcal{E}_{i/j,t}$ represents either IoT-node or edge encryption, $EncNR_{i,t}$ and $EncER_{j,t}$, respectively, $\mathcal{T}_{i/j,t}$ represents IoT-node or edge test, $NT_{i,t}$ and $ET_{j,t}$, respectively, λ is the security parameter, and $\epsilon(\lambda)$ is a negligible function.

Theorem 6 *PRF-SAKE achieves mutual authenticity under Definition 15.*

Proof: We prove Theorem 6 by considering the following two IoT-node and edge impersonations scenarios:

1. Case 1- Impersonating IoT-node: Suppose that $\mathcal{A}$ is able to obtain the network seed, e.g., by compromising an IoT-node in the network. In order to impersonate another IoT-node with pseudo identity $PID_{i,t}$, $\mathcal{A}$ should create a $(EncNR^{\mathcal{A}}_{i,t}, NT^{\mathcal{A}}_{i,t})$ pair such that it passes the edge authentication test shown in Section 6.4. Specifically, $NT^{\mathcal{A}}_{i,t} \stackrel{?}{=} NT_{i,t}$ where $NT^{\mathcal{A}}_{i,t} = (TS \| PID_{i,t} \| nR^{\mathcal{A}}_{i,t} \| AK_{i,t})$ and $EncNR^{\mathcal{A}}_{i,t} = nR^{\mathcal{A}}_{i,t} \oplus h(TS \| AK_{i,t} \| Seed)$. To this end, $\mathcal{A}$ has the following options:

   (a) Suppose that $\mathcal{A}$ is able to obtain the edge delivery key $DK_{j,t}$ at time t. Having the current edge delivery key $DK_{j,t}$, $\mathcal{A}$ needs to break the one-way PRF function given in Equation 6.1 to obtain the current authentication key $AK_{i,t}$ in order to create a matching $(EncNR^{\mathcal{A}}_{i,t}, NT^{\mathcal{A}}_{i,t})$ pair. In this option, $\mathcal{A}'s$ advantage is $= Adv_f(\mathcal{A})$.

   (b) $\mathcal{A}$ obtains $AK_{i,t}$ by exhaustive search to create matching $(EncNR^{\mathcal{A}}_{i,t}, NT^{\mathcal{A}}_{i,t})$ pair. The probability of $\mathcal{A}'s$ success is $= \frac{1}{2^{l_a}}$ where l_a is the length of $AK_{i,t}$.

2. Case 2- Impersonating Edge: Similar to case 1, suppose that $\mathcal{A}$ is able to obtain the network seed, by comprising an IoT-node in the network, and is able to leak the edge delivery key $DK_{j,t}$ at time t. In order to impersonate an edge, $\mathcal{A}$ should create a $(EncER^{\mathcal{A}}_{j,t}, EncDK^{\mathcal{A}}_{j,t}, ET^{\mathcal{A}}_{j,t})$ tuple such that it passes the IoT-node authentication test shown in Section 6.4. Specifically, $ET^{\mathcal{A}}_{j,t} \stackrel{?}{=} ET_{j,t}$ where $ET^{\mathcal{A}}_{i,t} = (h(TS\|PID_{i,t}\|eR^{\mathcal{A}}_{j,t}\|DK^{\mathcal{A}}_{j,t}\|SK^{\mathcal{A}}_{i,t}))$, $EncDK^{\mathcal{A}}_{j,t} = DK^{\mathcal{A}}_{j,t} \oplus h(TS\|DK_{j,(t-1)}\|Seed)$, and $EncER^{\mathcal{A}}_{j,t} = eR^{\mathcal{A}}_{j,t} \oplus h(TS\|AK_{i,t})$. To this end, $\mathcal{A}$ has the following options:

   (a) Having the current edge delivery key $DK_{j,t}$, $\mathcal{A}$ needs to break the one-way PRF function given in Equation 6.1 to obtain the current authentication key $AK_{i,t}$ in order to decrypt the current $EncNR_{i,t}$ to retrieve $nR_{i,t}$. Once the adversary has $nR_{i,t}$, he derives his own matching $(EncER^{\mathcal{A}}_{j,t}, EncDK^{\mathcal{A}}_{j,t}, ET^{\mathcal{A}}_{j,t})$ shown in Section 6.4. In this option, the adversary's advantage is $= Adv_f(\mathcal{A})$.

   (b) $\mathcal{A}$ obtains $AK_{i,t}$ by exhaustive search in order to decrypt the current $EncNR_{i,t}$ to retrieve $nR_{i,t}$, and hence, derives his own matching $(EncER^{\mathcal{A}}_{j,t}, EncDK^{\mathcal{A}}_{j,t}, ET^{\mathcal{A}}_{j,t})$. The probability of $\mathcal{A}'s$ success is $= \frac{1}{2^{l_a}}$ where l_a is the length of $AK_{i,t}$.

From the above two cases, it follows that the adversary's advantage in impersonating an IoT-node or edge is:

$$Adv^{auth}_{PRF\text{-}SAKE} \leq (\tfrac{1}{2^{l_a}} + Adv_f(\mathcal{A}))$$

■

Definition 16 (Forward & Backward Secrecy of *PRF-SAKE*) *We say that thePRF-SAKE protocol provides forward and backward message secrecy if for any given time t, and any probabilistic-polynomial-time adversary $\mathcal{A}$ that has records of previously encrypted network messages, $\mathcal{A}$ cannot learn previous or future session keys SK_l where $l \neq t$ if the long-term key or the current session key, Seed or SK_t, respectively, are exposed. Formally,*

$$Pr[SK^{\mathcal{A}}_{i,l} \leftarrow \mathcal{A}(1^{\lambda}, SK_{i,t}, Seed) : Sk^{\mathcal{A}}_{i,l} = SK_{i,l} \wedge (l \neq t)] \leq \epsilon(\lambda)$$

where λ is the security parameter, and $\epsilon(\lambda)$ is a negligible function.

Theorem 7 *PRF-SAKE achieves forward and backward message secrecy under Definition 16.*

Proof: We prove Theorem 7 by examining a scenario where $\mathcal{A}$ is able to leak both the current session key and the network seed, $SK_{i,t}$ and $Seed$, respectively. To reveal old or future session keys, $KS_{i,l}$ where $l \neq t$, $\mathcal{A}$ has the following options:

- $\mathcal{A}$ obtains the authentication key $AK_{i,l}$ by exhaustive search in order to decrypt $EncNR_{i,l}$ and $EncER_{j,l}$ shown in Section 6.4, to obtain $nR_{i,l}$ and $eR_{j,l}$, respectively, in order to derive the session key given in Equation 6.2, Section 6.4. Consequently, $\mathcal{A}'s$ advantage is $= \frac{1}{2^{l_a}}$.
- $\mathcal{A}$ obtains $SK_{i,l}$ by exhaustive search where the probability of $\mathcal{A}'s$ success is $= \frac{1}{2^{l_s}}$ where l_s is the length of session key.

It follows that the adversary's advantage in obtaining old or future session keys is $Adv^{fs}_{PRF\text{-}SAKE} \leq (\frac{1}{2^{l_s}})$

■

Definition 17 (Anonymity of *PRF-SAKE*) *We say that the PRF-SAKE protocol provides anonymity with respect to any probabilistic-polynomial-time external adversary $\mathcal{A}$ that has access to all network messages if $\mathcal{A}$ cannot link any message at any given time, t, to the IoT-node originator. Formally,*

$$Pr[PID^{\mathcal{A}}_{i,l} \leftarrow \mathcal{A}(1^{\lambda}, PID_{i,t}) \colon PID^{\mathcal{A}}_{i,l} = PID_{i,l} \wedge (l \neq t)] \leq \epsilon(\lambda)$$

where λ is the security parameter, and $\epsilon(\lambda)$ is a negligible function.

Theorem 8 *PRF-SAKE achieves anonymity with respect to any external adversary under Definition 17.*

Proof: We prove Theorem 8 by assuming that the adversary $\mathcal{A}$ has access to all messages exchanged in the network. $\mathcal{A}$ cannot link a particular message to previous or future IoT-node messages, because in each new session, the IoT-node generates a new pseudo-random ID using the sequence counter, $PID_{i,t} = h(Cntr_i \| PID_{i,(t-1)})$ as shown in Section 6.4- Equation 6.3. Consequently, the only option that $\mathcal{A}$ has to link IoT-node messages is to brute force the counter $Cntr_{i,t}$. Thus, the adversary's advantage in linking messages is $Adv^{link}_{PRF\text{-}SAKE} \leq (\frac{1}{2^{lc}})$ where c is the length of $Cntr_i$.

■

6.6 Protocol Evaluation

In this section, we evaluate our *PRF-SAKE* protocol in terms of run-time, storage requirements, and communication overhead. The computational run-time for the edge-IoT mutual authentication and key agreement phase is evaluated as follows. In each new session, both the IoT device and the edge compute $2\times(5\ hash{+}1\ PRF{+}1\ RNG{+}4\ XOR)$ where we consider a 1 PRF evaluation to be equivalent to one encryption operation. For memory storage, the IoT device needs to store five temporary values, and they are, the counter $Cntr$, pseudo-identity PID, authentication key AK, and derivation key DK. Additionally, there is a long-term key seed shared between each IoT device and the edge. Assuming all secrets are 256 $bits = 32\ Bytes$, we require a total of $5 \times 32 = 160\ Bytes$ memory space for each IoT device. On the other hand, the edge is required to store the same set of temporary keys for each node. Assuming N nodes owned by a particular edge, the total storage requirements for one edge $= (N \times 128) + 32\ Bytes$. The communication overhead in the authentication and key agreement phase is computed for the following two packets, $Pkt_1 = ID_{edgej}, < TS \| PID_{i,t} \| EncNR_{i,t} \| NT_{i,t} >$ and $Pkt_2 = PID_{i,t}, < TS \| EncER_{j,t} \| EncDK_{j,t} \| ET_{j,t} >$. Assuming $TS\ size = 4\ Bytes$, $ID\ size = 3\ Bytes$, $SHA\text{-}256 = 32\ Bytes$, and all other values to be 32 $Bytes$, the communication overhead

is $(3 + 2 \times 4 + 3 \times 32 + 2 \times 32) = 171\ Bytes$.

Table 6.1: Performance Comparisons Authentication Phase

Ref	Description	Com. Overhead	Storage Req.	Computations	Benchmark ✧
Proposed	- Symmetric authentication key agreement (SAKE) - IoT & edge	171 *Bytes*	160 *Bytes*	5 *hash* + 1 *PRF evl* + 1 *RNG* + 4 *XOR*	$(5\times0.451+1\times0.226+1\times1.0) = 3.481\ msec$
Ibrahim [76]	- Mutual authentication - IoT & Fog sever	*NA*	32 *Bytes*	1 *hash* + 1 *sym enc.* + 1 *sym dec.*	$(1 \times 0.451 + 1 \times 0.226 + 1 \times 0.226) = 0.903\ msec$
Wang [179]	- Mutual authentication - Computation offloading - IoT & edge	*NA*	56 *Bytes* ♦	15 *hash*+2 *MAC*	$(15 \times 0.451 + 2 \times 0.902) = 8.569\ msec$
Aman [9]	- Mutual authentication key agreement - IoT & edge	17 *Bytes* ✪	42 *Bytes* ✪	2 *hash* + 3 *MAC* + 2 *sym enc* + 2 *PUF*	$(2\times0.451+3\times0.902+2\times0.226+2\times1.0) = 6.06\ msec$
Gope [67]	- Mutual authentication key agreement - IoT & edge	17 *Bytes* ✪	*NA*	5 *hashes*+2 *PUF*	$5 \times 0.451 + 2 \times 1.0 = 4.255\ msec$

NA: Not available in the publication.
✧: Based on ARM Cortex-M0 48MHz ATECC508A HW accelerated. AES 16 Bytes, Hash 32 Bytes, and RGN 32 Bytes. Also, based on HMAC $\sim$ 2 Hash
♦: based on secrets= 16 Bytes
✪: Based on longest message, and ID= $1Byte$, $MAC = 4Bytes$, $Nonce = 6Bytes$. Based on $< C, R >= 16Bytes$. Based on ideal PUF. Also, based on $PUF = RNG$.

Comparative Evaluation: Up to our knowledge, no authentication protocols to support offloading for static three-tier cloud-edge-IoT has been proposed in the literature before. However, we compare our *PRF-SAKE* protocol to other symmetric edge protocols such as [76, 179]. Additionally, we include similar lightweight protocols such as [9, 67]. The performance comparisons are shown in a Table 6.1, and the achieved security goals comparisons are shown in Table 6.2. In Ibrahim protocol [76], the fog-user and fog-server achieve mutual authentication; however, the protocol uses real identities for both fog-server and fog-user. Thus, the attacker can trace the communication to both user and server. Additionally, the protocol relies on having a long-term key between the fog-user and the for-server which if leaked all previous and future session keys will be leaked. Thus, forward and backward secrecy are not achieved in [76]. In terms of performance, Ibrahim's protocol requires the fog-server to store $N \times 32\ Bytes$, assuming the long-term shared key is 32 *Bytes*. On the other hand, the fog-user is required to store only the current session key and the long-term shared key, 32 *Bytes*.

Wang *et al.* [179], is a mutual authentication scheme with light computation requirements. The objective of the scheme is to compute the offloading charges; thus, the forward and backward secrecy are not applicable in the protocol. Nevertheless, the protocol achieves anonymity and untraceability with respect to the attacker. In [9, 67], the authors present mutual authentication protocols which require the installation of PUFs inside the IoT devices. Both protocols are lightweight and achieve mutual authentication between servers and IoT devices, anonymity with respect to any external adversary, and forward secrecy. However, both schemes do not provide backward secrecy. Furthermore, the PUF are subject to noise and a special hardware which can be expensive. On the other hand, our *PRF-SAKE* protocol does not require the installation of PUF devices and achieves mutual authentication, anonymity with respect to any external adversary, forward secrecy, and backward secrecy using symmetric key cryptography with light storage requirements and low communication overhead as shown in Tables 6.1 and 6.2.

Table 6.2: Security Goals Comparisons

Security Goals	**Proposed**	**Ibrahim [76]**	**Wang [179]**	**Aman [9]**	**Gope [67]**
Mutual authentication	✓	✓	✓	✓	✓
Anonymity	✓	✗	✓	✓	✓
Trace-ability	✓	✗	✓	✓	✓
Forward secrecy	✓	✗	*NA*	✓	✓
Backward secrecy	✓	✗	*NA*	✗	✗

6.7 Summary

We proposed a lightweight symmetric-key authentication and key agreement protocol for edge computing offloading applications. In our protocol, the IoT devices and edge entities are static and enrolled in the network upon registration in which the IoT devices are assigned to their edge owner based on geographical proximity. Our protocol is designed specifically for IoT-edge computational and storage offloading from IoT device

to edge. We analyzed the security of the protocol and showed that it provides anonymity with respect to external adversaries, forward secrecy, and backward secrecy. We also showed that our protocol is lightweight when compared to other protocols.

Chapter 7

Conclusion and Future Work

7.1 Summary

Security is one of the major concerns in the edge computing paradigm. First, the de-centralization process of having several distributed servers instead of one central server increases the chances of security attacks. Second, in all three layers of the edge computing paradigm, wireless communication is used which opens the door for various attacks such as eavesdropping, intercepting messages, injecting messages, reply attacks, or DoS attacks. The generic PKC security solutions are not suitable for the edge computing paradigm, especially because the majority of IoT devices are characterized by small processors, confined memory space, and low battery lifetime. To this end, we proposed several lightweight security protocols, and in what follows, we give a brief summary of the contribution accomplished in this thesis.

In Chapter 2, we presented the architecture of EC, the different related de-centralized paradigms such fog computing and mobile edge computing, and highlighted their differences. We presented the EC security model which includes threat model, security properties, security primitives, and analysis tools. We also presented background information on the utilized security primitives used in our proposed protocols.

In Chapter 3, we proposed a lightweight broadcasting authentication protocol for edge computing applications. Specifically, the edge entity broadcast messages using session keys which are derived securely from a hash function. The protocol utilizes hash chains and authenticated encryption which makes it resilient to quantum attacks. Moreover, entities are not required to hold a permanent master key, and all session keys are derived securely from a hash function. As a use case, we presented a smart emergency system where an edge application broadcasts alert messages for individual responder groups when specific events occur. We formally defined and proved the main security properties of our protocol and compared it to other lightweight protocols in terms of security and performance.

In Chapter 4, we proposed a lightweight group authentication protocol with a session key-agreement. Most of the previously proposed 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-MAC, is lightweight and provides multiple asynchronous authentications. Furthermore, we implemented a simple key refreshing mechanism in which, in each session, a new session-key between an IoT-node and the authenticating server is established without the need for re-distributing new shares. In our security analysis, we showed that our protocol provides group authentication, message forward secrecy, and prevents several attacks. Additionally, we presented a formal automated verification using Verifpal tool and showed that our scheme has better performance than other relative schemes in terms of communication complexity, secret-share re-distribution, and session key derivations.

In Chapter 5, we exploited the advantages of two security primitives, physically unclonable functions and Shamir's secret sharing scheme to design a lightweight group authentication scheme for edge-computing applications. Specifically, we applied PUFs on SSS and utilize the SSS-homomorphic property to achieve multiple-time group-authentications with the same set of shares. Our *PUF-GAS* scheme is lightweight, establishes a new group

key-agreement per session and supports efficient node-evicting mechanism. Furthermore, in *PUF-GAS*, the group nodes do not store any shares; instead, the nodes derive their secret-shares from their PUF-responses. We formally analyzed our protocol theoretically and with AVISPA to show that our scheme achieves message secrecy and authenticity. Additionally, we evaluated our scheme in terms of storage, run-time, and communication overhead. We also presented a comparative evaluation of our scheme with others in terms of security and performance.

In Chapter 6, we proposed a lightweight symmetric key pseudo-random-function-based protocol which provides edge-IoT mutual authentication, forward secrecy, backward secrecy, and anonymity. We proved the security goals of the protocol and compared it to other lightweight authentication protocols. Also, we showed that the computational complexity of our protocol is only 1 pseudo-random function evaluation, 1 random number generation, 5 hash, and 4 xor operations.

7.2 Future Work

Further in-depth studies and security solutions for the EC paradigm are needed. Indeed, it is anticipated that the next few years will witness a change in the edge computing architecture, framework, and its applications. Specifically, researchers are taking advantage of the unprecedented breakthroughs in artificial intelligence (AI), machine learning (ML), deep learning (DL) models and algorithms and utilize them in the EC-paradigm. This resulted a merge between edge computing and artificial intelligence referred to Edge Intelligence (EI) [204]. Deng *et al.* put forward two classifications for the usage of AI in edge computing, AI for Edge and AI on Edge [55]. The former method utilizes AI algorithms to grant edge more intelligence perhaps to solve optimization problems or improve performance. On the other hand, AI on Edge applies training and inference of AI models and handling massive generated data on edge servers. Indeed, both of these

methods increase the need for security analysis and investigations. Finally, new security technologies have been developing, and further in-depth studies and security analysis are needed to verify their feasibility and application on EC, blockchains (BC) as an example. In this section, we shed light on the changes in the EC-paradigm that can extend the contribution of this thesis.

- **Intelligence-enabled Edge Computing:** AI-for-Edge is a classification in which the AI, ML, or DL models provide the technology to find optimal solutions to key EC problems. For instance, the EC offloading or resource allocation problem in different layers can be solved or improved by applying AI algorithms and models. Additionally, the new EI-paradigm creates many application-scenarios ranging from personal applications to large governmental sectors. For example, smart city applications, personalized assistants, video surveillance, and other applications implement AI to improve employees' productivity and raise the standards of living. However, the fast EI-development creates several security concerns, and in-depth security studies and analysis are required for its applications and frameworks.

- **Intelligence on Edge (big data and mass secure solutions):** The proliferation of smart devices, mobile computing, and IoT made EC very efficient de-centralized solution. Indeed, processing data at the edge of the network is more efficient than traversing it to the mega-scale cloud datacenters. The fast adoption of AI supported applications concurrently with the billions of edge data bytes production tremendously increase the demand on edge data processing and optimization [55]. However, the AI implementation on edge creates several novel application scenarios, security concerns, performance issues, and novel training and inference implementation frameworks. For example, Zhou *et al.* propose several training and inference models from cloud-edge co-inference (level 1) to on-device (level 6) [204]. On the other hand, the introduction of Federated Learning by McMahan *et al.* [119] allows the distribution of machine model training among local devices while protecting

against data leakage. With Federated Learning several security techniques can be applied to protect shared data; for example, secure multiparty computations, zero-knowledge proof, homomorphic encryption, and differential privacy.

On the other hand, a common scenario in the future EI-paradigm is having large number of AI nodes, i.e. large groups, with large volume of data in which groups may disseminate the big data or share training. Although we have proposed two group authentication protocols in our thesis, but more mass secure solutions for AI nodes that address a wide spectrum of security goals need to be investigated. Finally, most of the EI devices are lightweight and accessible to attackers, and this creates big security concerns especially for the node-theft attacks. Thus, security solutions must address these threats and mitigate them.

- **Blockchain:** BC technology is currently being integrated in many EC smart applications such smart grid, autonomous driving, mobile edge computing, vehicle-to-grid, and others. Additionally, many smart applications utilize BC to meet authentication, access control, key management protocol requirements, and to prevent several denial-of-service attacks. However, BC comes with two main trade-offs, security with efficiency and transparency with privacy. Additionally, many BC-based solutions are not suitable for the low-end devices. For example, Zero-Knowledge-Proofs are not suitable for the low-end devices. Thus, further security analysis and investigations are needed for this newly introduced technology.

Bibliography

[1] D. Abbasinezhad-Mood and M. Nikooghadam. An ultra-lightweight and secure scheme for communications of smart meters and neighborhood gateways by utilization of an arm cortex-m microcontroller. *IEEE Transactions on Smart Grid*, 9(6): 6194–6205, 2017.

[2] D. Abbasinezhad-Mood and M. Nikooghadam. Efficient anonymous password-authenticated key exchange protocol to read isolated smart meters by utilization of extended chebyshev chaotic maps. *IEEE Transactions on Industrial Informatics*, 14(11):4815–4828, 2018.

[3] D. Abbasinezhad-Mood, A. Ostad-Sharif, S. M. Mazinani, and M. Nikooghadam. Provably-secure escrow-less chebyshev chaotic map-based key agreement protocol for vehicle to grid connections with privacy protection. *IEEE Transactions on Industrial Informatics*, 2020.

[4] M. Abdalla and D. Pointcheval. Interactive diffie-hellman assumptions with applications to password-based authentication. In A. S. Patrick and M. Yung, editors, *Financial Cryptography and Data Security*, pages 341–356. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-31680-0.

[5] A. Ahmed and E. Ahmed. A survey on mobile edge computing. In *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, pages 1–8. ISCO, 2016.

[6] M. A. Aleisa, A. Abuhussein, and F. T. Sheldon. Access control in fog computing: Challenges and research agenda. *IEEE Access*, 8:83986–83999, 2020.

[7] Alloy. https://www.alloy.com/, Nov. Accessed: 2021.

[8] 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 of Things Journal*, 8(6):4004–4022, 2021.

[9] M. N. Aman, K. C. Chua, and B. Sikdar. Mutual authentication in iot systems using physical unclonable functions. *IEEE Internet of Things Journal*, 4(5):1327–1340, 2017.

[10] M. N. Aman, U. Javaid, and B. Sikdar. A privacy-preserving and scalable authentication protocol for the internet of vehicles. *IEEE Internet of Things Journal*, 8(2): 1123–1139, 2021.

[11] A. B. Amor, M. Abid, and A. Meddeb. A privacy-preserving authentication scheme in an edge-fog environment. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 1225–1231. IEEE, 2017.

[12] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. Real-time video analytics: The killer app for edge computing. *Computer*, 50(10):58–67, 2017.

[13] S. Annigeri and S. Raut. Tinyecc. https://github.com/ShubhamAnnigeri/tinyECC-ArduinoIDE, Accessed: 2023.

[14] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, et al. The avispa tool for the automated validation of internet security protocols and applications. In

International conference on computer aided verification, pages 281–285. Springer, 2005.

[15] A. Armando and L. Compagna. Sat-based model-checking for security protocols analysis. *International Journal of Information Security*, 7(1):3–32, 2008.

[16] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[17] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE transactions on information theory*, 29(2):208–210, 1983.

[18] AVISPA. Automated validation of internet security protocols and applications. http://www.avispa-project.org/, July Accessed: 2021.

[19] G. Avoine, S. Canard, and L. Ferreira. Symmetric-key authenticated key exchange (sake) with perfect forward secrecy. In *Cryptographers' Track at the RSA Conference*, pages 199–224. Springer, 2020.

[20] Y. Aydin, G. K. Kurt, E. Ozdemir, and H. Yanikomeroglu. A flexible and lightweight group authentication scheme. *IEEE Internet of Things Journal*, 2020.

[21] Y. Aydin, G. K. Kurt, E. Ozdemir, and H. Yanikomeroglu. Group authentication for drone swarms. In *2021 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, pages 72–77. IEEE, 2021.

[22] G. Bansal and B. Sikdar. Location aware clustering: Scalable authentication protocol for uav swarms. *IEEE Networking Letters*, 3(4):177–180, 2021.

[23] G. Bansal and B. Sikdar. Fault resilient authentication architecture for drone networks. In *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 866–871. IEEE, 2022.

[24] C. Baritel-Ruet. Formal security proofs of cryptographic : A necessity achieved using easycrypt. https://tel.archives-ouvertes.fr/tel-03177617/document, Accessed: 2022.

[25] B. Barras, S. Boutin, C. Cornes, J. Courant, J.-C. Filliatre, E. Gimenez, H. Herbelin, G. Huet, C. Munoz, C. Murthy, et al. *The Coq proof assistant reference manual Version 6.1*. PhD thesis, Inria, 1997.

[26] D. Basin, S. Mödersheim, and L. Vigano. An on-the-fly model-checker for security protocol analysis. In *European Symposium on Research in Computer Security*, pages 253–270. Springer, 2003.

[27] S. Basudan. Lega: A lightweight and efficient group authentication protocol for massive machine type communication in 5g networks. *Journal of Communications and Information Networks*, 5(4):457–466, 2020.

[28] J. C. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Conference on the theory and application of cryptographic techniques*, pages 251–260. Springer, 1986.

[29] Berkeley. http://people.eecs.berkeley.edu/ culler/papers/mica-sensors.pdf, Nov. Accessed: 2019.

[30] G. R. Blakley. Safeguarding cryptographic keys. In *Managing Requirements Knowledge, International Workshop on*, page 313. IEEE Computer Society, 1979.

[31] B. Blanchet. Security protocol verification: Symbolic and computational models. In P. Degano and J. D. Guttman, editors, *Principles of Security and Trust*, pages 3–29. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-28641-4.

[32] B. Blanchet. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. *Privacy and Security*, 1(1-2):1 – 135, Oct. 2016.

[33] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Annual international cryptology conference*, pages 41–55. Springer, 2004.

[34] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 416–432. Springer, 2003.

[35] D. Boneh and V. Shoup. A graduate course in applied cryptography, 2020.

[36] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. acm, 2012.

[37] C. Boyd, A. Mathuria, and D. Stebila. *Protocols for authentication and key establishment*, volume 1. Springer, 2003.

[38] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 453–474. Springer, 2001.

[39] 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 of Things Journal*, 7(6):5329–5344, 2020.

[40] 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 of Things Journal*, 6 (2):1561–1575, 2018.

[41] K. Cao, S. Hu, Y. Shi, A. Colombo, S. Karnouskos, and X. Li. A survey on edge and edge-cloud computing assisted cyber-physical systems. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2021.

[42] X. Cao, W. Kou, L. Dang, and B. Zhao. Imbas: Identity-based multi-user broadcast authentication in wireless sensor networks. *Computer communications*, 31(4):659–667, 2008.

[43] C.-C. Chang and H.-D. Le. A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks. *IEEE Transactions on wireless communications*, 15(1):357–366, 2015.

[44] S.-M. Chang, S. Shieh, W. W. Lin, and C.-M. Hsieh. An efficient broadcast authentication scheme in wireless sensor networks. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 311–320. the 2006 ACM symposium on information, computer and communications security, 2006.

[45] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay. A PUF-based secure communication protocol for iot. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(3):1–25, 2017.

[46] S. A. Chaudhry, H. Alhakami, A. Baz, and F. Al-Turjman. Securing demand response management: A certificate-based access control in smart grid edge computing infrastructure. *IEEE Access*, 8:101235–101243, 2020.

[47] D. Chaum and E. Van Heyst. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 257–265. Springer, 1991.

[48] 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 of Things Journal*, pages 1–1, 2021.

[49] Y. Chen, J.-F. Martínez, P. Castillejo, and L. López. An anonymous authentication and key establish scheme for smart grid: Fauth. *Energies*, 10(9):1354, 2017.

[50] 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*, 8:71833–71839, 2020.

[51] H.-Y. Chien. Group authentication with multiple trials and multiple authentications. *Security and Communication Networks*, 2017, 2017.

[52] H.-Y. Chien. Group-oriented range-bound key agreement for internet of things scenarios. *IEEE Internet of Things Journal*, 5(3):1890–1903, 2018.

[53] H.-Y. Chien, J.-K. Jan, and Y.-M. Tseng. A practical (t, n) multi-secret sharing scheme. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 83(12):2762–2765, 2000.

[54] J. Cui, X. Chen, J. Zhang, Q. Zhang, and H. Zhong. Towards achieving fine-grained access control of data in connected and autonomous vehicles. *IEEE Internet of Things Journal*, 2020.

[55] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, 7(8):7457–7469, 2020.

[56] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 523–540. Springer, 2004.

[57] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[58] P. Dong, W. Wang, X. Shi, and T. Qin. Lightweight key management for group communication in body area networks through physical unclonable functions. In *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pages 102–107. IEEE, 2017.

[59] M. A. Ferrag, A. Derhab, L. Maglaras, M. Mukherjee, and H. Janicke. Privacy-preserving schemes for fog-based iot applications: Threat models, solutions, and challenges. In *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, pages 37–42. SaCoNet, 2018.

[60] M. M. Fouda, Z. M. Fadlullah, N. Kato, R. Lu, and X. S. Shen. A lightweight message authentication scheme for smart grid communications. *IEEE Transactions on Smart grid*, 2(4):675–685, 2011.

[61] Y. Gao, S. F. Al-Sarawi, and D. Abbott. Physical unclonable functions. *Nature Electronics*, 3(2):81–91, 2020.

[62] Y. Gao, Y. Su, L. Xu, and D. C. Ranasinghe. Lightweight (reverse) fuzzy extractor with multiple reference puf responses. *IEEE Transactions on Information Forensics and Security*, 14(7):1887–1901, 2019.

[63] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 148–160. IEEE, 2002.

[64] H. Gilbert and H. Handschuh. Security analysis of sha-256 and sisters. In *International workshop on selected areas in cryptography*, pages 175–193. Springer, 2003.

[65] I. Goldberg, D. Stebila, and B. Ustaoglu. Anonymity and one-way authentication in key exchange protocols. *Designs, Codes and Cryptography*, 67(2):245–269, 2013.

[66] P. Gope, J. Lee, and T. Q. Quek. Lightweight and practical anonymous authentication protocol for rfid systems using physically unclonable functions. *IEEE Transactions on Information Forensics and Security*, 13(11):2831–2843, 2018.

[67] P. Gope and B. Sikdar. Lightweight and privacy-preserving two-factor authentication scheme for iot devices. *IEEE Internet of Things Journal*, 6(1):580–589, 2018.

[68] Y. Guo, Z. Zhang, and Y. Guo. Fog-centric authenticated key agreement scheme without trusted parties. *IEEE Systems Journal*, pages 1–10, 2020.

[69] B. Haase and B. Labrique. Aucpace: Efficient verifier-based pake protocol tailored for the iiot. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 1–48, 2019.

[70] D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[71] L. Harn. Group authentication. *IEEE Transactions on computers*, 62(9):1893–1898, 2012.

[72] M. U. Hassan, M. H. Rehmani, and J. Chen. Differential privacy techniques for cyber physical systems: A survey. *IEEE Communications Surveys Tutorials*, 22(1): 746–789, 2020.

[73] J. He and E. Dawson. Multistage secret sharing based on one-way function. *Electronics Letters*, 30(19):1591–1592, 1994.

[74] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[75] M. Huang, B. Yu, and S. Li. Puf-assisted group key distribution scheme for software-defined wireless sensor networks. *IEEE Communications Letters*, 22(2):404–407, 2018.

[76] M. H. Ibrahim. Octopus: An edge-fog mutual authentication scheme. *IJ Network Security*, 18(6):1089–1101, 2016.

[77] S. Iftene. General secret sharing based on the chinese remainder theorem with applications in e-voting. *Electronic Notes in Theoretical Computer Science*, 186: 67–84, 2007.

[78] A. Irshad, S. A. Chaudhry, O. A. Alomari, K. Yahya, and N. Kumar. A novel pairing-free lightweight authentication protocol for mobile cloud computing framework. *IEEE Systems Journal*, 2020.

[79] S. Jangirala, A. K. Das, and A. V. Vasilakos. Designing secure lightweight blockchain-enabled rfid-based authentication protocol for supply chains in 5g mobile edge computing environment. *IEEE Transactions on Industrial Informatics*, 16(11): 7081–7093, 2020.

[80] 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 Systems Journal*, 14(1):560–571, 2020.

[81] B. Jiang, J. Li, G. Yue, and H. Song. Differential privacy for industrial internet of things: Opportunities, applications, and challenges. *IEEE Internet of Things Journal*, 8(13):10430–10451, 2021.

[82] H. Jiang, J. Pei, D. Yu, J. Yu, B. Gong, and X. Cheng. Applications of differential privacy in social network analysis: A survey. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.

[83] J. Katz and A. Y. Lindell. Aggregate message authentication codes. In *Cryptographers' Track at the RSA Conference*, pages 155–169. Springer, 2008.

[84] J. Katz and Y. Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[85] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues, and M. Guizani. Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay. *IEEE Communications Magazine*, 56(2):44–51, 2018.

[86] K. Kaur, S. Garg, G. Kaddoum, M. Guizani, and D. N. K. Jayakody. A lightweight and privacy-preserving authentication protocol for mobile edge computing. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[87] K. Kaya and A. A. Selçuk. Threshold cryptography based on asmuth–bloom secret sharing. *Information sciences*, 177(19):4148–4160, 2007.

[88] M. N. Khan, A. Rao, and S. Camtepe. Lightweight cryptographic protocols for iot-constrained devices: A survey. *IEEE Internet of Things Journal*, 8(6):4132–4156, 2021.

[89] K. Kifayat, M. Merabti, Q. Shi, and D. Llewellyn-Jones. An efficient multi-parameter group leader selection scheme for wireless sensor networks. In *2009 International Conference on Network and Service Security*, pages 1–5. IEEE, 2009.

[90] H. Kim, E. Kang, E. A. Lee, and D. Broman. A toolkit for construction of authorization service infrastructure for the internet of things. In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pages 147–158. the second international conference on Internet-of-Things design and implementation, 2017.

[91] H. Kim and E. A. Lee. Authentication and authorization for the internet of things. *IT Professional*, 19(5):27–33, 2017.

[92] H. Kim, A. Wasicek, B. Mehne, and E. A. Lee. A secure network architecture for the internet of things based on local authorization entities. In *2016 IEEE 4th*

International Conference on Future Internet of Things and Cloud (FiCloud), pages 114–122. IEEE, 2016.

[93] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, CCS '00, page 235–244. Association for Computing Machinery, New York, NY, USA, 2000. ISBN 1581132034. URL https://doi.org/10.1145/352600.352638.

[94] N. Kobeissi, G. Nicolas, and M. Tiwari. Verifpal: cryptographic protocol analysis for the real world. In *International Conference on Cryptology in India*, pages 151–202. Springer, 2020.

[95] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. ISBN 978-3-540-48405-9.

[96] H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication, 1997.

[97] 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 *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 832–837. IEEE, 2013.

[98] 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. *Computer Networks*, 57(17): 3492–3510, 2013.

[99] 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. *Computer Networks*, 99:66–81, 2016.

[100] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.

[101] L. Lamport. The temporal logic of actions. *ACM Trans. Program. Lang. Syst.*, 16 (3):872–923, 1994.

[102] T.-F. Lee. Provably secure anonymous single-sign-on authentication mechanisms using extended chebyshev chaotic maps for distributed computer networks. *IEEE Systems Journal*, 12(2):1499–1505, 2015.

[103] T.-F. Lee, X. Ye, and S.-H. Lin. Anonymous dynamic group authenticated key agreements using physical unclonable functions for internet of medical things. *IEEE Internet of Things Journal*, pages 1–1, 2022.

[104] H. Li, R. Lu, L. Zhou, B. Yang, and X. Shen. An efficient merkle-tree-based authentication scheme for smart grid. *IEEE Systems Journal*, 8(2):655–663, 2013.

[105] 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 of Things Journal*, 3(3):408–417, 2015.

[106] Q. Li and G. Cao. Multicast authentication in the smart grid with one-time signature. *IEEE Transactions on Smart Grid*, 2(4):686–696, 2011.

[107] S. Li, I. Doh, and K. Chae. A group authentication scheme based on lagrange interpolation polynomial. In *2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pages 386–391. IEEE, 2016.

[108] X. Li, D. Li, J. Wan, C. Liu, and M. Imran. Adaptive transmission optimization in sdn-based industrial internet of things with edge computing. *IEEE Internet of Things Journal*, 5(3):1351–1360, 2018.

[109] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie. Toward edge-based deep learning in industrial internet of things. *IEEE Internet of Things Journal*, 7(5): 4329–4341, 2020.

[110] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, 2017.

[111] D. Liu and P. Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. Technical report, North Carolina State University. Dept. of Computer Science, 2002.

[112] D. Liu and P. Ning. Multilevel μtesla: Broadcast authentication for distributed sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 3 (4):800–836, 2004.

[113] D. Liu, P. Ning, S. Zhu, and S. Jajodia. Practical broadcast authentication in sensor networks. In *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 118–129. IEEE, 2005.

[114] N. Liu, J. Chen, L. Zhu, J. Zhang, and Y. He. A key management scheme for secure communications of advanced metering infrastructure in smart grid. *IEEE Transactions on Industrial electronics*, 60(10):4746–4756, 2012.

[115] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8): 1697–1716, 2019.

[116] R. Maes, A. Van Herrewege, and I. Verbauwhede. Pufky: A fully functional puf-based cryptographic key generator. In *Cryptographic Hardware and Embedded Systems–CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings 14*, pages 302–319. Springer, 2012.

[117] M. H. Mahalat, D. Karmakar, A. Mondal, and B. Sen. PUF based secure and lightweight authentication and key-sharing scheme for wireless sensor network. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 18(1):1–23, 2021.

[118] P. Mall, R. Amin, A. K. Das, M. T. Leung, and K.-K. R. Choo. Puf-based authentication and key agreement protocols for iot, wsns, and smart grids: A comprehensive survey. *IEEE Internet of Things Journal*, 9(11):8205–8228, 2022.

[119] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[120] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran, and S. Guizani. Internet-of-things-based smart cities: Recent advances and challenges. *IEEE Communications Magazine*, 55(9):16–24, 2017.

[121] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 1997.

[122] O. Millwood, J. Miskelly, B. Yang, P. Gope, E. Kavun, and C. Lin. Puf-phenotype: A robust and noise-resilient approach to aid intra-group-based authentication with dram-pufs using machine learning. *arXiv preprint arXiv:2207.04692*, 2022.

[123] M. D. Mitzenmacher and A. Perrig. Bounds and improvements for biba signature schemes. *Harvard*, 2002.

[124] M. B. Mollah, J. Zhao, D. Niyato, Y. L. Guan, C. Yuen, S. Sun, K.-Y. Lam, and L. H. Koh. Blockchain for the internet of vehicles towards intelligent transportation systems: A survey. *IEEE Internet of Things Journal*, 8(6):4157–4185, 2021.

[125] A. Mosenia and N. K. Jha. A comprehensive study of security of internet-of-things. *IEEE Transactions on emerging topics in computing*, 5(4):586–602, 2016.

[126] M. Nakkar, R. Altawy, and A. Youssef. Lightweight broadcast authentication protocol for edge-based applications. *IEEE Internet of Things Journal*, 7(12):11766–11777, 2020.

[127] M. Nakkar, R. AlTawy, and A. Youssef. Lightweight authentication and key agreement protocol for edge computing applications. In *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, pages 415–420. IEEE, 2021.

[128] M. Nakkar, R. AlTawy, and A. Youssef. GASE: A lightweight group authentication scheme with key agreement for edge computing applications. *IEEE Internet of Things Journal*, pages 1–1, 2022.

[129] M. Nakkar, R. AlTawy, and A. Youssef. Lightweight group authentication scheme leveraging shamir's secret sharing and pufs. *IEEE Transactions on Network Science and Engineering*, pages 1–17, 2024.

[130] M. Nakkar, M. Mahmoud, and A. Youssef. Fault analysis-resistant implementation of rainbow signature scheme. In *2017 29th International Conference on Microelectronics (ICM)*, pages 1–5. IEEE, 2017.

[131] W. Neumann. Horse: an extension of an r-time signature scheme with fast signing and verification. In *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, volume 1, pages 129–134 Vol.1. ITCC, 2004.

[132] NIST. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf, Nov. Accessed: 2019.

[133] V. Odelu, A. K. Das, M. Wazid, and M. Conti. Provably secure authenticated key agreement scheme for smart grid. *IEEE Transactions on Smart Grid*, 9(3): 1900–1910, 2016.

[134] 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 of Things Journal*, 6(3):4118–4149, 2018.

[135] OpenFogConsortium. Openfog reference architecture for fog computing. https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf, September 2021.

[136] L.-J. Pang and Y.-M. Wang. A new (t, n) multi-secret sharing scheme based on shamir's secret sharing. *Applied Mathematics and Computation*, 167(2):840–848, 2005.

[137] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The tesla broadcast authentication protocol. *Rsa Cryptobytes*, 5(2):2–13, 2002.

[138] J. Pieprzyk, H. Wang, and C. Xing. Multiple-time signature schemes against adaptive chosen message attacks. In *International Workshop on Selected Areas in Cryptography*, pages 88–100. Springer, 2003.

[139] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb. Survey on multi-access edge computing for internet of things realization. *IEEE Communications Surveys Tutorials*, 20(4):2961–2991, 2018.

[140] P. Porambage, C. Schmitt, P. Kumar, A. Gurtov, and M. Ylianttila. Two-phase authentication protocol for wireless sensor networks in distributed iot applications. In *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2728–2733. Ieee, 2014.

[141] ProVerif. https://bblanche.gitlabpages.inria.fr/proverif/, Nov. Accessed: 2021.

[142] C. Pu and Y. Li. Lightweight authentication protocol for unmanned aerial vehicles using physical unclonable function and chaotic system. In *2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN*, pages 1–6. IEEE, 2020.

[143] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu. Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Communications Surveys Tutorials*, 22(4):2462–2488, 2020.

[144] Y. Qiu and M. Ma. Secure group mobility support for 6lowpan networks. *IEEE Internet of Things Journal*, 5(2):1131–1141, 2018.

[145] Raspberry-Pi. https://www.raspberrypi.com/products/raspberry-pi-3-model-b/, Nov. Accessed: 2022.

[146] P. Remlein, M. Rogacki, and U. Stachowiak. Tamarin software the tool for protocols verification security. In *2020 Baltic URSI Symposium (URSI)*, pages 118–123. IEEE, 2020.

[147] K. Ren, W. Lou, K. Zeng, and P. J. Moran. On broadcast authentication in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 6(11):4136–4144, 2007.

[148] K. Ren, S. Yu, W. Lou, and Y. Zhang. Multi-user broadcast authentication in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 58(8):4554–4564, 2009.

[149] 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 of Things Journal*, pages 1–1, 2021.

[150] L. Reyzin and N. Reyzin. Better than biba: Short one-time signatures with fast signing and verifying. In *Australasian Conference on Information Security and Privacy*, pages 144–153. Springer, 2002.

[151] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *Fast Software Encryption: 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004. Revised Papers 11*, pages 371–388. Springer, 2004.

[152] S. Roy, S. Chatterjee, A. K. Das, S. Chattopadhyay, S. Kumari, and M. Jo. Chaotic map-based anonymous user authentication scheme with user biometrics and fuzzy extractor for crowdsourcing internet of things. *IEEE Internet of Things Journal*, 5 (4):2884–2895, 2017.

[153] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 237–249. ACM, 2010.

[154] Scyther. https://people.cispa.io/cas.cremers/scyther/, Nov. Accessed: 2021.

[155] M. Seifelnasr, R. AlTawy, and A. Youssef. Efficient inter-cloud authentication and micropayment protocol for iot edge computing. *IEEE Transactions on Network and Service Management*, 18(4):4420–4433, 2021.

[156] M. Seifelnasr, M. Nakkar, A. Youssef, and R. AlTawy. A lightweight authentication and inter-cloud payment protocol for edge computing. In *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, pages 1–4. IEEE, 2020.

[157] A. Setyawan Sajim. Open-source software-based sram-puf for secure data and key storage using off-the-shelf sram. *Thesis*, 2018.

[158] K. Sha, T. A. Yang, W. Wei, and S. Davari. A survey of edge computing-based designs for iot security. *Digital Communications and Networks*, 6(2):195–202, 2020.

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

[160] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[161] J. Shao and Z. Cao. A new efficient (t, n) verifiable multi-secret sharing (vmss) based on ych scheme. *Applied Mathematics and Computation*, 168(1):135–140, 2005.

[162] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[163] W. Shi and S. Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[164] K.-A. Shim, Y.-R. Lee, and C.-M. Park. Eibas: An efficient identity-based broadcast authentication scheme in wireless sensor networks. *Ad Hoc Networks*, 11(1):182–189, 2013.

[165] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[166] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptol. ePrint Arch.*, 2004:332, 2004.

[167] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734, 2018.

[168] A. H. Sodhro, S. Pirbhulal, and V. H. C. de Albuquerque. Artificial intelligence-driven mechanism for edge computing-based industrial applications. *IEEE Transactions on Industrial Informatics*, 15(7):4235–4243, 2019.

[169] F. Song. A note on quantum security for post-quantum cryptography. In *International Workshop on Post-Quantum Cryptography*, pages 246–265. Springer, 2014.

[170] SPAN+AVISPA. Span, the security protocol animator for avispa. http://people.irisa.fr/Thomas.Genet/span/, July Accessed: 2021.

[171] Tamarin. http://tamarin-prover.github.io/, Nov. Accessed: 2021.

[172] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni. A systematic survey of industrial internet of things security: Requirements and fog computing opportunities. *IEEE Communications Surveys Tutorials*, 22(4):2489–2520, 2020.

[173] M. Tanveer, A. H. Zahid, M. Ahmad, A. Baz, and H. Alhakami. Lake-iod: Lightweight authenticated key exchange protocol for the internet of drone environment. *IEEE Access*, 8:155645–155659, 2020.

[174] Z. Tian, Y. Wang, Y. Sun, and J. Qiu. Location privacy challenges in mobile edge computing: Classification and exploration. *IEEE Network*, 34(2):52–56, 2020.

[175] M. Turuani. The cl-atse protocol analyser. In *International Conference on Rewriting Techniques and Applications*, pages 277–286. Springer, 2006.

[176] A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann. Reverse fuzzy extractors: Enabling lightweight mutual authentication for puf-enabled rfids. In *Financial Cryptography and Data Security: 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers 16*, pages 374–389. Springer, 2012.

[177] D. Von Oheimb. The high-level protocol specification language hlpsl developed in the eu project avispa. In *Proceedings of APPSEM 2005 workshop*, pages 1–17. APPSEM, 2005.

[178] J. R. Wallrabenstein. Practical and secure iot device authentication using physical unclonable functions. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 99–106. IEEE, 2016.

[179] F. Wang, Y. Xu, L. Zhu, X. Du, and M. Guizani. Lamanco: A lightweight anonymous mutual authentication scheme for n-times computing offloading in iot. *IEEE Internet of Things Journal*, 6(3):4462–4471, 2018.

[180] Q. Wang, H. Khurana, Y. Huang, and K. Nahrstedt. Time valid one-time signature for time-critical multicast data authentication. In *IEEE INFOCOM 2009*, pages 1233–1241. IEEE, 2009.

[181] M. Wazid, P. Bagga, A. K. Das, S. Shetty, J. J. Rodrigues, and Y. H. Park. Akmiov: Authenticated key management protocol in fog computing-based internet of vehicles deployment. *IEEE Internet of Things Journal*, 6(5):8804–8817, 2019.

[182] M. Wazid, A. K. Das, N. Kumar, V. Odelu, A. Goutham Reddy, K. Park, and Y. Park. Design of lightweight authentication and key agreement protocol for vehicular ad hoc networks. *IEEE Access*, 5:14966–14980, 2017.

[183] M. Wazid, A. K. Das, V. Odelu, N. Kumar, M. Conti, and M. Jo. Design of secure user authenticated key management protocol for generic iot networks. *IEEE Internet of Things Journal*, 5(1):269–282, 2017.

[184] M. Wazid, A. K. Das, V. Odelu, N. Kumar, and W. Susilo. Secure remote user authenticated key establishment protocol for smart home environment. *IEEE Transactions on Dependable and Secure Computing*, 17(2):391–406, 2020.

[185] R. Weatherley. Arduino cryptography library. https://rweather.github.io/arduinolibs/crypto.html, Accessed: 2023.

[186] WolfSSL. https://www.wolfssl.com/docs/benchmarks, Nov. Accessed: 2019.

[187] T.-Y. Wu, Z. Lee, M. S. Obaidat, S. Kumari, S. Kumar, and C.-M. Chen. An authenticated key exchange protocol for multi-server architecture in 5g networks. *IEEE Access*, 8:28096–28108, 2020.

[188] Y. Xia, R. Qi, S. Ji, J. Shen, T. Miao, and H. Wang. Puf-assisted lightweight group authentication and key agreement protocol in smart home. *Wireless Communications and Mobile Computing*, 2022, 2022.

[189] C. Xu, J. Ren, D. Zhang, and Y. Zhang. Distilling at the edge: A local differential privacy obfuscation framework for iot data analytics. *IEEE Communications Magazine*, 56(8):20–25, 2018.

[190] A. Yang, J. Weng, K. Yang, C. Huang, and X. Shen. Delegating authentication to edge: A decentralized authentication architecture for vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15, 2020.

[191] C.-C. Yang, T.-Y. Chang, and M.-S. Hwang. A (t, n) multi-secret sharing scheme. *Applied Mathematics and Computation*, 151(2):483–490, 2004.

[192] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289–330, 2019.

[193] S. Yu, A. K. Das, Y. Park, and P. Lorenz. Slap-iod: Secure and lightweight authentication protocol using physical unclonable functions for internet of drones in smart city environments. *IEEE Transactions on Vehicular Technology*, pages 1–15, 2022.

[194] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang. A survey on the edge computing for the internet of things. *IEEE access*, 6:6900–6919, 2017.

[195] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang. A survey on the edge computing for the internet of things. *IEEE Access*, 6:6900–6919, 2018.

[196] H. Yıldız, M. Cenk, and E. Onur. Plgakd: A PUF-based lightweight group authentication and key distribution protocol. *IEEE Internet of Things Journal*, 8(7): 5682–5696, 2021.

[197] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[198] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu. Data security and privacy-preserving in edge computing paradigm: Survey and open issues. *IEEE Access*, 6: 18209–18237, 2018.

[199] J. Zhang, H. Zhong, J. Cui, M. Tian, Y. Xu, and L. Liu. Edge computing-based privacy-preserving authentication framework and protocol for 5g-enabled vehicular networks. *IEEE Transactions on Vehicular Technology*, 69(7):7940–7954, 2020.

[200] P. Zhang, J. K. Liu, F. R. Yu, M. Sookhak, M. H. Au, and X. Luo. A survey on access control in fog computing. *IEEE Communications Magazine*, 56(2):144–149, 2018.

[201] J. Zhao, J. Zhang, and R. Zhao. A practical verifiable multi-secret sharing scheme. *Computer Standards & Interfaces*, 29(1):138–141, 2007.

[202] P. Zhao, G. Zhang, S. Wan, G. Liu, and T. Umer. A survey of local differential privacy for securing internet of vehicles. *The Journal of Supercomputing*, 76(11): 8391–8412, 2020.

[203] Y. Zheng, W. Liu, C. Gu, and C.-H. Chang. Puf-based mutual authentication and key exchange protocol for peer-to-peer iot applications. *IEEE Transactions on Dependable and Secure Computing*, pages 1–18, 2022.

[204] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

[205] Z. Zhou, B. Wang, M. Dong, and K. Ota. Secure and efficient vehicle-to-grid energy trading in cyber physical systems: Integration of blockchain and edge computing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):43–57, 2020.

[206] S. Zhu, S. Setia, and S. Jajodia. Leap+ efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(4): 500–528, 2006.