Authentication Protocols for IoT Edge Computing

Mohamed Ibrahieem

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

August 2024

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mohamed Mahdy Seifelnasr Ibrahieem**

Entitled: **Authentication Protocols for IoT 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. Dongyu Qiu

_____ External Examiner
Dr. Huapeng Wu

_____ External to Program
Dr. Abdelwahab Hamou-Lhadj

_____ Examiner
Dr. Walter Lucia

_____ Examiner
Dr. Mohammad Mannan

_____ Supervisor
Dr. Amr M. Youssef

Approved by _____
Dr. Jun Yan, Graduate Program Director

August 8^{th}, 2024
Date of Defence

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

Abstract

Authentication Protocols for IoT Edge Computing

Mohamed Ibrahieem, Ph.D.

Concordia University, 2024

The proliferation of IoT has led to vast interconnectivity, generating massive data that exceeds the processing capabilities of IoT devices. Traditional IoT-cloud models, where devices offload computations to centralized cloud servers, are increasingly inadequate due to the expected surge in IoT devices, projected to surpass 75 billion by 2025. This growth intensifies cloud vulnerability to single points of failure and highlights the need for alternatives that meet QoS requirements like low latency and location awareness. The 3-tier IoT-edge-cloud architecture offers a solution by processing data at nearby edge nodes, improving location awareness, and mitigating single-point-of-failure. While this distributed approach meets QoS requirements, it introduces security challenges, such as offloading data to distributed edge nodes without prior registration. Additionally, an adversary can trace the edge node attached to the IoT device and compromise the privacy of an IoT device user. Moreover, many deployed IoT devices are vulnerable to hardware compromise and unauthorized access, raising significant privacy and security concerns that hinder the broader adoption of edge computing.

In this thesis, we address the above challenges by proposing efficient and secure authentication protocols for IoT applications in edge computing. Our proposed protocols include Symmetric Key Authentication with Forward Secrecy (SKAFS), Symmetric Key

Inter-Cloud Authentication and Redeemable Micropayment Protocol (SKICAP), Mutual Authentication Privacy-Preserving Protocol with Forward Secrecy (MAPFS), and Conditional Privacy-Preserving Message Authentication for VANET Emergency Exchange (CP-MAVE). The proposed protocols utilize lightweight cryptographic primitives to realize efficient protocols for edge computing. Moreover, the proposed protocols fulfill the security requirements for IoT applications, such as IoT device anonymity, session unlinkability, and resilience to hardware compromise of IoT devices.

For our proposed protocols, we provided formal security analyses based on computationally hard problems. Furthermore, we evaluated their performance in terms of communication overhead and computational complexity and compared them with other closely related protocols. Finally, we implemented prototypes of our proposed protocols using socket programming, simulating the message flow between the protocol entities to calculate their end-to-end latency and confirm the efficiency of our proposed protocols. The proliferation of IoT has led to vast interconnectivity, generating massive data that exceeds the processing capabilities of IoT devices. Traditional IoT-cloud models, where devices offload computations to centralized cloud servers, are increasingly inadequate due to the expected surge in IoT devices, projected to surpass 75 billion by 2025. This growth intensifies cloud vulnerability to single points of failure and highlights the need for alternatives that meet Quality of Service (QoS) requirements like low latency and location awareness. The 3-tier IoT-edge-cloud architecture offers a solution by processing data at nearby edge nodes, improving location awareness, and mitigating single-point-of-failure. While this distributed approach meets QoS requirements, it introduces security challenges, such as offloading data to distributed edge nodes without prior registration. Additionally, an adversary can trace the edge node attached to the IoT device and compromise the privacy of an IoT device user. Moreover, with 2.38 billion IoT devices vulnerable to hardware compromise and unauthorized access, raising significant privacy and security concerns that hinder the broader adoption of edge computing. In this thesis, we address the above challenges by proposing efficient and secure authentication protocols for IoT applications in edge computing. Our proposed protocols include Symmetric Key Authentication with Forward Secrecy (SKAFS), Symmetric Key Inter-Cloud Au-

thentication and Redeemable Micropayment Protocol (SKICAP), Mutual Authentication Privacy-Preserving Protocol with Forward Secrecy (MAPFS), and Conditional Privacy-Preserving Message Authentication for VANET Emergency Exchange (CP-MAVE). The proposed protocols utilize lightweight cryptographic primitives to realize efficient protocols for edge computing. Moreover, the proposed protocols fulfill the security requirements for IoT applications, such as IoT device anonymity, session unlinkability, and resilience to hardware compromise of IoT devices. For our proposed protocols, we provided formal security analyses based on computationally hard problems. Furthermore, we evaluated their performance in terms of communication overhead and computational complexity and compared them with other closely related protocols. Finally, we implemented prototypes of our proposed protocols using socket programming, simulating the message flow between the protocol entities to calculate their end-to-end latency and confirm the efficiency of our proposed protocols.

Acknowledgments

First and foremost, I express my gratitude to Allah (God), the Almighty, the Most Merciful, and the Most Gracious, for granting me the strength to pursue my aspirations. Without my faith in Him, this thesis would not have been accomplished.

I would like to express my sincere gratitude and appreciation to my supervisor, Dr. Amr Youssef. I am always indebted to him for his continuous support, encouragement, patience, guidance, and immense knowledge. I have learned a lot from him, and I am very proud to have been his student.

In addition, I would like to extend my sincere appreciation to the examining committee: Dr. Huapeng Wu, Dr. Hamou-Lhadj, Dr. Walter Lucia, and Dr. Mohammad Mannan, for their helpful insights, valuable discussions, and guidance throughout each milestone of the Ph.D. journey.

I express my heartfelt gratitude to the Fonds de recherche du Quebec – Nature et Technologies (FRQNT) for funding my doctoral studies, which relieved financial burdens and enabled focused research, especially in the last year of my Ph.D.

Furthermore, I am deeply grateful for the unwavering support of my friends here in Canada: Dr. Mohamed Elhattab, Dr. Mohamed Bayoumi, Louis Pierre, Iing Muttakhiroh, and Mohamed Abdelwahab, throughout my Ph.D. journey. Their encouragement, guidance, and camaraderie have been invaluable, helping me navigate the challenges and celebrate the triumphs along the way. Their friendship has truly enriched my experience, and I am forever thankful for their presence in my life.

Last, but not least, to my beloved parents, especially my dear father who is no longer with us: your love, support, and sacrifices have guided me, especially during my Ph.D. journey. Mom, your strength is my inspiration. And to my siblings, especially

Esraa, your love has made every step easier. Thank you for shaping who I am today.

Mohamed Ibrahieem

Table of Contents

List of Figures **xiii**

List of Tables **xv**

List of Acronyms **xvi**

Chapter 1 Introduction **1**

1.1 Overview 1

1.2 Motivation 2

1.3 Contributions 3

1.4 Thesis Outline 6

Chapter 2 Background **7**

2.1 Edge Computing 7

2.1.1 Overview 7

2.1.2 Applications 9

2.1.3 Security Goals 10

2.1.4 Attacks and Vulnerabilities 12

2.1.5 Mutual Authentication Protocols for Edge Computing 13

2.2 Notation 16

2.3 Cryptographic Primitives 17

2.3.1 Physical Unclonable Function 17

2.3.2 Merkle Tree 21

2.3.3 Elliptic Curve 21

2.3.4 Schnorr Zero-Knowledge Proof [121] . . . 23

Chapter 3 Symmetric Key Authentication Protocol with Forward Secrecy 24

3.1 Introduction . . . 24

3.2 System Model and Threat model . . . 27

3.2.1 System Model . . . 27

3.2.2 Threat Model . . . 28

3.3 Protocol Design . . . 28

3.3.1 Registration Phase . . . 29

3.3.2 Mutual Authentication Phase . . . 30

3.3.3 Key Agreement Phase . . . 31

3.3.4 Message Exchange Phase . . . 33

3.3.5 Dynamic IoT device Addition . . . 34

3.4 Soundness and Formal Security Analysis . . . 34

3.4.1 Soundness of SKAFS . . . 34

3.4.2 Security Analysis of SKAFS using AVISPA . . . 37

3.4.3 Formal Security Analysis . . . 38

3.5 Performance Analysis and Comparison . . . 47

3.5.1 Communication Cost . . . 47

3.5.2 Storage Requirements . . . 48

3.5.3 Computation Cost . . . 49

3.5.4 Performance Benchmarking and Comparison . . . 50

3.6 Summary . . . 54

Chapter 4 Symmetric Key Inter-Cloud Authentication and Redeemable Micropayment Protocol 55

4.1 Introduction . . . 55

4.2 System Model, Entities, and Design Goals . . . 57

4.2.1 System Model . . . 57

4.2.2 Threat Model . . . 59

4.2.3 Design Goals . . . 59

4.3 Protocol Design and Implementation . . . 60
4.3.1 Protocol Overview . . . 61
4.3.2 Revocation of gateways . . . 66
4.4 Security Analysis . . . 67
4.4.1 Adversary Model . . . 67
4.4.2 Security Analysis . . . 68
4.4.3 SKICAP Freshness, Guaranteed Redemption, and Traceability . . . 73
4.5 Performance Evaluation . . . 74
4.5.1 Computation Overhead . . . 75
4.5.2 Execution Time . . . 76
4.5.3 Communication Overhead . . . 77
4.5.4 Storage Overhead . . . 78
4.6 Summary . . . 78

Chapter 5 Mutual Authentication Privacy-Preserving Protocol with Forward Secrecy for the IoT-edge-cloud 80
5.1 Introduction . . . 80
5.2 System Model and Threat Model . . . 82
5.2.1 System Model . . . 83
5.2.2 Threat Model . . . 84
5.3 Protocol Design . . . 84
5.3.1 Setup Phase . . . 85
5.3.2 Registration Phase . . . 85
5.3.3 Mutual Authentication and Key Agreement Phase . . . 88
5.3.4 Revoking the anonymity of misbehaving IoTs . . . 89
5.4 Security Analysis . . . 90
5.4.1 Adversary Model . . . 90
5.4.2 Security Analysis . . . 92
5.4.3 MAPFS Freshness, Anonymity, Backward Secrecy, and Forward Secrecy Properties. . . . 97

5.5 Performance Evaluation . . . 99
5.5.1 Storage Requirements . . . 101
5.5.2 Computation Cost . . . 101
5.5.3 Communication Overhead . . . 101
5.5.4 Execution Time . . . 102
5.6 Summary . . . 104

Chapter 6 Conditional Privacy-preserving Message Authentication Protocol for VANET Emergency Message Exchange 105

6.1 Introduction . . . 105
6.2 System Model and Threat Model . . . 108
6.2.1 System Model . . . 108
6.2.2 Threat Model . . . 109
6.2.3 Security Requirements . . . 110
6.3 Protocol Design . . . 110
6.3.1 Setup Phase . . . 111
6.3.2 ECA Registration with distributed KGCs . . . 111
6.3.3 Vehicle's Signature Generation Phase . . . 112
6.3.4 Message Sending Phase . . . 113
6.3.5 Traceability of the Sender Vehicle ECA Domain . . . 114
6.4 Security Analysis . . . 115
6.4.1 Security Definitions . . . 115
6.4.2 Formal Security Analysis . . . 117
6.4.3 Unlinkability . . . 119
6.4.4 Conditional Privacy Preservation . . . 120
6.4.5 Resistance to Replay and Modification Attacks . . . 120
6.5 Performance Evaluation . . . 121
6.5.1 Computation Cost . . . 121
6.5.2 Communication Overhead . . . 122
6.5.3 Execution Time . . . 123

6.6 Summary 124

Chapter 7 Conclusion and Future Work **125**

7.1 Conclusion 125

7.2 Future Work 127

Bibliography **128**

List of Figures

2.1 The IoT-edge-cloud paradigm 8

2.2 An example illustrating the Merkle proof for element $c \in MT$ which consists of the nodes H_d and H_{ab} 21

3.1 IoT device registration 30

3.2 IoT gateway registration 30

3.3 Mutual authentication between the IoT device and the IoT gateway for the i^{th} session 32

3.4 Key agreement between the IoT device and the IoT gateway for the i^{th} session 33

3.5 Obtaining the CO service offered by the IoT gateway 34

3.6 AVISPA results for the specified goals (i.e., the secrecy of the session key, the secrecy of IoT ID, and authentication of the IoT device, the IoT gateway, and the CA) 39

3.7 The cryptographic overhead time for protocols [10], [64], [123], [65], [27], [28], and SKAFS. The time for the IoT device and the gateway are computed on Raspberry Pi 1 and Raspberry Pi 4, respectively 49

3.8 Communication cost of [10], [64], [123], [65], [27], [28], [159] and SKAFS . . 52

3.9 The scaling of the storage required at CA per IoT device with the size of the synchronization set 53

4.1 An illustration of the considered IoT-edge-cloud paradigm. The dashed lines represent the transition of an IoT user from the subscribed home cloud coverage area, depicted in green, to the hosting cloud coverage area, depicted in red . . . 58
4.2 An example of a tree of secrets for eight IoT devices pseudo-identities . . . 62
4.3 Merkle tree of the tips of a hash chain of eight IoT devices . . . 63
4.4 Registration of the IoT device, IoT_{010} . . . 63
4.5 Registration of the IoT gateway G_v . . . 63
4.6 Mutual authentication between IoT_{010} and the IoT-Gateway G_v of CA_v . . . 64
4.7 Charges redemption for claimed token X' in return for n' CO units from the hosting cloud . . . 66
4.8 IoT device, IoT gateway and total execution time of [15], [79], [109], [142], [29] and SKICAP . . . 77

5.1 MAPFS system model . . . 83
5.2 The IoT gateway registration . . . 87
5.3 The IoT device registration . . . 87
5.4 Mutual authentication phase . . . 89
5.5 Evaluations of the overhead associated with cryptographic primitives in protocols [84], [80], [81], [94], [82], [32] and MAPFS . . . 103

6.1 VANET Architecture . . . 108
6.2 ECA registration . . . 112
6.3 Vehicle obtaining signature from ECA . . . 113
6.4 Sending a message from the vehicle to the TMA . . . 114
6.5 Evaluations of cryptographic primitives overhead time . . . 123

List of Tables

3.1 Security/ Functionality properties compared to other IoT-edge-cloud schemes . . 26
3.2 The evolution of the internal states in case $D_{IG} = 0$ 37
3.3 The evolution of the internal states in case $D_{IG} = -1$ 37
3.4 Storage, Computation and Communication Cost 47
3.5 Average cryptographic overhead time (msec) 51
3.6 Performance comparison based on the computation complexity 51

4.1 Comparison with other related IoT-Edge-Cloud protocols. 57
4.2 Protocol storage, computation and communication requirements 76
4.3 Execution time of various cryptographic operations 77

5.1 Comparison with other related protocols. 82
5.2 Summary of our performance analysis 100
5.3 Average cryptographic overhead time using 1000 runs 100
5.4 Performance comparison based on the computation complexity 102

6.1 Comparison with other message authentication protocols for VANETs. 107
6.2 Average cryptographic overhead time using 1000 runs 121
6.3 Performance comparison based on the computation complexity 122

List of Acronyms

AES	Advanced Encryption Standard
CA	Cloud Admin
CCA	Centralized Certificate Authority
CK	Canetti-Krawczyk
CL-AtSe	Constraint Logic based Attack Searcher
CL-PKC	Certificate-Less Public-Key Cryptography
CO	Computation Offloading
CRP	Challenge-Response Pair
DLP	Discrete Log Problem
DPUF	DRAM-based PUF
DRAM	Dynamic Random Access Memory
DY	Dolev-Yao
ECC	Elliptic Curve Cryptography
ECA	Enterprise Certificate Authority
ECDH	Elliptic Curve Diffie-Hellman
ECDLP	Elliptic Curve Discrete Log Problem
ECDDHP	Elliptic-Curve Decisional Diffie-Hellman Problem
ETNO	European telecommunications network operators' association
FHD	Fractional Hamming Distance
HD	Hamming Distance

HLPSL	High-Level Protocol Specification Language
IF	Intermediate Format
IoMT	Internet of Medical Things
IoT	Internet of Things
IoV	Internet of Vehicles
KGC	Key Generation Center
MA	Mutual Authentication
MI	Message Integrity
OBUs	On-Board Units
OF	Output Format
OFMC	On-the-fly Model-Checker
PKI	Public-Key Infrastructure
PPT	Probabilistic Polynomial Time
PUF	Physical Unclonable Function
QoS	Quality of Service
RSUs	Road Side Units
SATMC	SAT-based Model-Checker
SS	Semantic Security
TA	Trusted Authority
TA4SP	Tree Automata based on Automatic Approximations for the Analysis of Security Protocols
TTP	Trusted Third Party
TMAs	Traffic Management Agencies
VANET	Vehicular Ad-Hoc Network
ZKP	Zero-Knowledge Proof

Chapter 1

Introduction

1.1 Overview

The Internet of Things (IoT) stands as the backbone framework of smart cities, where many IoT devices with Internet connectivity are installed everywhere for sensing, transmitting, processing, receiving, and actuating with minimal human intervention. IoT devices are deployed in many applications such as smart grids, smart homes, smart transportation systems, smart healthcare monitoring, and smart emergency services, which shape our future. It is expected that by 2025, the number of IoT devices will exceed 75 billion [4]. Consequently, the amount of data requiring processing on the cloud is expected to increase tremendously. This expanding trend of IoT devices and their widespread nature impose some Quality of Service (QoS) requirements that the conventional IoT-cloud architecture cannot fulfill. These requirements include low latency, location awareness, support for mobility, and geo-distribution [53]. With such an expanding trend, the single point of failure, in the traditional IoT-cloud, becomes more critical.

In this context, the 3-tier IoT-edge-cloud architecture has been considered [26]; limited-resource IoT devices and sensor nodes transmit their sensed data to more computation-capable edge nodes (e.g., access points or base transceiver stations) in their proximity for

data processing and storage [111]. Such a non-centralized architecture offers location awareness and mobility support and can satisfy the required IoT application QoS.

However, the adoption of edge computing requires proper security and privacy-preserving mechanisms. Edge computing inherits security and privacy issues from cloud computing. Moreover, it suffers from other threats, such as man-in-the-middle attacks, due to the adoption of the middle edge nodes for collecting and processing IoT users' personal information. Consequently, mutual authentication between IoT devices and edge nodes is essential to ensure security, allowing or restricting users' access to the system. Mutual authentication prevents illegitimate users from enjoying the service. In the conventional IoT-cloud paradigm, IoT users store their data on cloud servers, which are trusted domains for IoT users. In edge computing, as IoT users and edge nodes are in different trust domains. This model differs from the conventional IoT-cloud model and mutual authentication protocols for IoT-cloud cannot be applied directly to the IoT-edge-cloud paradigm. Furthermore, adopting the new IoT-edge-cloud paradigm faces more challenges as the privacy of an IoT device user can be compromised when an external adversary traces the location of the edge node paired with the IoT device. Additionally, the deployed sensor nodes are vulnerable to physical compromise attacks; according to Gartner statistics on the IoT market segment in 2020 [3], a total of 2.38 billion IoT devices are used in unprotected public sectors, e.g., transportation, retail and wholesale trade, natural resources, and utilities.

1.2 Motivation

Throughout this thesis, we address the challenges above in adopting edge computing by providing efficient mutual authentication protocols using cryptographic primitives to meet the security and QoS requirements in IoT applications for edge computing. Specifically, we aim to 1) thwart the vulnerability of the IoT user credentials deployed in public

areas; 2) enable mutual authentication between IoT devices and edge nodes without requiring an online Cloud Admin (CA) during the authentication process; and 3) develop a privacy-preserving mutual authentication protocol, maintaining unlinkability between IoT device sessions and edge nodes, and preventing further location traceability by external adversaries.

1.3 Contributions

Our research in the Ph.D. program focuses on developing new authentication protocols suitable for the emerging IoT-edge-cloud paradigm. Anonymous authentication protocols for edge computing should prevent illegitimate endpoint devices from accessing the computation service by ensuring that only legitimate endpoints are authenticated [96]. Additionally, the traceability of misbehaving endpoints should be maintained. Our research has resulted in the development of four protocols.

In the first protocol, we enable the mutual authentication between the low endpoints IoT devices and edge nodes to benefit from the Computation Offloading (CO) service of distributed edge nodes. We introduce the Symmetric Key Authentication protocol with Forward Secrecy (SKAFS). We tackle Desynchronization-based Denial-of-Service Attack (DDSA), where external adversaries disrupt the updating and synchronization of derivation keys among IoT devices and edge nodes of the service provider. SKAFS is resilient to physical compromise attacks and Physical Unclonable Function (PUF) modeling attacks. Moreover, it utilizes a shared long-term key for decoding the pseudo-identities of IoT devices, ensuring constant search complexity for identifying devices. Additionally, it embeds a synchronization scheme to ensure that the state-update process occurs at the beginning of each session, enhancing resilience to DDSA.

In the second protocol, we realize an efficient inter-cloud authentication and micropayment protocol for edge computing. We introduce Symmetric Key Inter-Cloud

Authentication and redeemable micropayment Protocol (SKICAP). The aim is to develop a secure delegation-based authentication protocol enabling roaming IoT devices to access CO services from foreign service providers. The protocol utilizes hash chains, Merkle trees, trees of secrets, and PUF. PUF enables IoT devices to compute authentication keys instead of digitally storing them, mitigating credential vulnerability due to hardware compromise of the IoT device. To ensure guaranteed charge redemption, the IoT device provides a payment token and Merkle membership proof to the hosting CA. The hosting CA verifies the payment token and Merkle tree proof to confirm the subscription of the requesting IoT device for CO service with the home CA.

In the third protocol, we allow the IoT-edge authentication without the need for an online CA during the authentication process. We present Mutual Authentication Privacy-preserving protocol with Forward Secrecy (MAPFS). The objective is to devise a scalable, privacy-preserving mutual authentication protocol for the IoT-edge-cloud paradigm, preventing external adversaries or malicious edge nodes from efficiently identifying or correlating incoming IoT requests. The protocol employs Diffie-Hellman key exchange, Schnorr Zero-Knowledge Proof (ZKP) of discrete log knowledge, and Schnorr signatures. The Diffie-Hellman protocol facilitates the establishment of a session key between IoT devices and edge nodes. Schnorr signatures authenticate the IoT device to the edge node and vice versa. The Schnorr ZKP is used to prove the knowledge of the secret that relates the randomized IoT authentication token to the protocol's public parameters.

In the fourth protocol, we enable the collabartive traceability of misbehaved vehicle. We introduce a Conditional Privacy-preserving Message Authentication protocol for Vehicular Ad-Hoc Network (VANET) Emergency message exchange (CP-MAVE). The objective is to enable a vehicle traveling to another city or country to obtain an authentication token from its enterprise cloud admin. Later on, the vehicle sends an authenticated traffic message regarding traffic conditions (e.g., car accidents, congestion, and fires) to the city's traffic management agency. The protocol eliminates the need for an online

CA during sending the message to the traffic management agency. The protocol utilizes Schnorr ZKP of discrete logarithm knowledge and Schnorr signatures. Distributed key generation centers are responsible for generating signing keys for the enterprise CA, and the traceability of misbehaving vehicles is maintained through collaboration among these centers. The protocol preserves the anonymity of the sending vehicle and ensures resilience against message replay and modification attacks.

It should be noted that the protocols presented are specific to each use case discussed in their respective chapters and are not intended for general application.

The contributions of this thesis can be summarized as follows:

- We introduce four protocols SKAFS, SKICAP, MAPFS, and CP-MAVE. SKAFS, and SKICAP are for realizing mutual authentication between IoT devices and edge nodes while MAPFS, and CP-MAVE are for anonymous message communication. The first three protocols, SKAFS, SKICAP, and MAPFS, appear in [123–125]. The proposed protocols utilize cryptographic primitives such as PUFs, hash functions, trees of secrets, Merkle trees, Schnorr zero-knowledge proof of discrete logarithm knowledge [121], and Schnorr signatures [120] to establish efficient authentication protocols meeting the required QoS for IoT applications.

- Under the assumption of computationally hard problems (i.e., the indistinguishability property of PUF as in SKICAP and SKAFS or the intractability of the elliptic curve discrete logarithm problem as in MAPFS and CP-MAVE), we provide formal security proofs for the presented protocols.

- We conduct a performance analysis of our protocols, focusing on their communication overhead, storage, and computation requirements. Additionally, we compare the execution time of these protocols with other closely related ones.

- We implement prototypes for the protocols using socket programming between the

protocol entities. This setup allows us to simulate the message flow between the protocol entities in a real-time experiment and calculate the end-to-end latency of the proposed protocol. The source code for the Python implementation is available in the GitHub repository at https://github.com/M-Mahdy-Seif.

Additional research conducted during this Ph.D but not included in this thesis has been published in [126, 127].

1.4 Thesis Outline

The remainder of the thesis is organized as follows: In Chapter 2, we provide an overview of edge computing, the notations used in the thesis, and background information on cryptographic primitives utilized in our protocols. Chapter 3 introduces our protocol SKAFS for IoT authentication with edge nodes. Chapter 4 presents SKICAP, a protocol for inter-cloud authentication. Chapter 5 presents MAPFS for IoT authentication with edge nodes without the need for an online CA. Chapter 6 introduces CP-MAVE, a protocol for inter-domain communications without reliance on an online CA. Finally, in Chapter 7, we conclude the work conducted throughout this thesis and present our possible future research directions.

Chapter 2

Background

In this chapter, we provide background on the edge computing paradigm, notations used throughout the thesis, and the cryptographic primitives that were utilized.

2.1 Edge Computing

In this section, we give an overview of the edge computing paradigm, some applications related to the edge computing paradigm, and the required design and security goals. Also, we give a literature review on the mutual authentication protocols for edge computing.

2.1.1 Overview

To mitigate the single point of failure in the conventional IoT-cloud paradigm and to meet the expected QoS requirements for the IoT applications, the three-tier IoT-edge-cloud paradigm is proposed. In such a paradigm, the middle edge layer is introduced between the low-end devices and the cloud [26]. In such a paradigm, resource-constrained IoT devices outsource heavy computations to a nearby edge node (e.g., routers, switches, and base stations), as depicted in Fig. 2.1, to avoid the high latency in the traditional

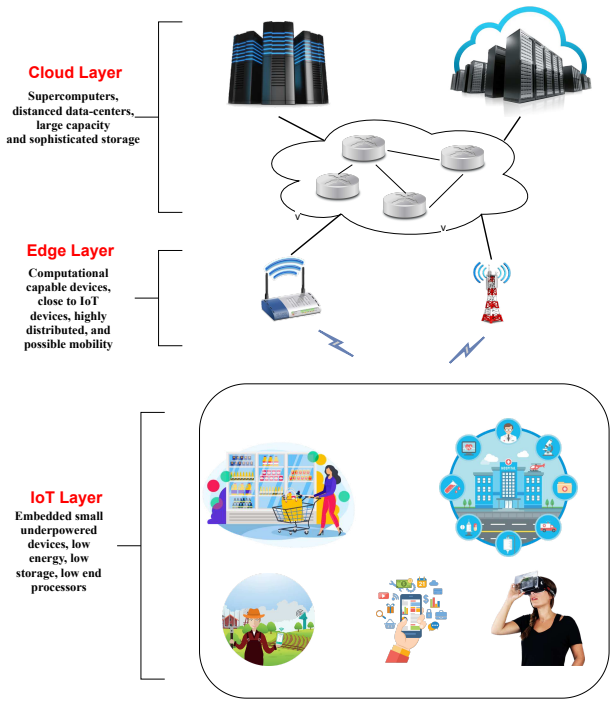

Figure 2.1: The IoT-edge-cloud paradigm

IoT-cloud framework where the data is sent to the distant cloud for further computation.

The main outstanding feature of the edge node is that it is closer to low-end devices than the fog node. As stated by the OpenFog Consortium [20], edge computing is defined as the processing of the data at the edge layer of the network closer to the end devices. The edge layer in the edge computing architecture involves servers, applications, and small clouds. Moreover, edge computing operates its edge nodes, and data are transferred back through the cloud to establish node-to-node traffic. The motivation of edge computing is that the computation and storage should occur close to the data source (IoT device). Initially, it was intended that edge computing should provide storage and computations only one-hop away from the data source. Practically, it could be more than one-hop. Unlike fog computing, edge computing is only limited to edges. Therefore, edge computing has lower latency than fog computing and cloud computing due to its location. In terms of service, edge computing has higher availability.

2.1.2 Applications

In what follows, we list some of the applications for edge computing.

1. Consumer IoT: the main target of this paradigm is the human end-user. Examples of consumer IoT devices are smartphones, tablets, and other electronic devices connected to the Internet. The communication is always machine-to-user communication such as smart payment, and smart conferencing [135].

2. Industrial IoT (IIoT): the main goal is the integration of the industrial manufacturing aspects with the business process. The motivation is to integrate all aspects of industrial manufacturing with the Information Technology (IT) that is associated with the business process. A typical example of an IIoT smart application is monitoring the production procedure and organizing autonomous industrial plants that operate without much human intervention [135]. Therefore, communication in IIoT is mainly machine-oriented unlike in the consumer IoT model where the communication is human-centered.

3. Smart Transportation: the main goal is to satisfy the traveler's needs and improve traffic efficiency either by improving the vehicle occupancy ratio or reducing the number of vehicles on the road [161]. Therefore, some schemes have been proposed that focus on ride-sharing among travelers by matching ride requests to both operating and idle vehicles [162] to minimize the total travel time distance or cost. Other schemes have been proposed that focus on reducing computational overhead to reduce the response time in matching vehicle-rider pairs and to reduce the number of operating vehicles while guaranteeing traffic efficiency at the same time [161].

4. Smart Agriculture: the terms smart farming, precision agriculture, and smart agriculture are used interchangeably in the literature [93]. In smart agriculture, new technologies are used to minimize the environmental impact and to maximize the

utilization of the resources. As an example, for monitoring the crop status, sensors are used, and based on the measurement values from the sensor, actuators manage agricultural processes related to animals, crops, greenhouses, irrigation, soil, and weather. This results in advancement in harvest forecasting, weather prediction, crop productivity, water conservation, real-time data collection, reduced operation costs, equipment monitoring, and accurate farm and field evaluation. In a smart agriculture system, agriculture data such as humidity, temperature, PH, and soil conditions are collected by heterogeneous data sources including robots, autonomous tractors, drones, harvesters, sensors, and actuators. Based on the sensed agriculture data different actuators such as water sprinklers, ventilation devices, lighting, automated windows (in glasshouses), and water nutrition pumps react. The number of cloud-based agricultural systems and physical systems is increasing to achieve a range of monitoring and analyzing objectives. Crop farming, livestock, and greenhouses are typical applications of edge computing. Some of these applications are implemented with the help of IoT-based sensors and devices by using wireless sensor networks (WSNs) [151].

2.1.3 Security Goals

In what follows, we list the required security goals and emphasize their importance for mutual authentication protocols in the IoT-edge-cloud paradigm.

- Mutual Authentication: the communication between low-end devices and the edge node is done through a wireless public channel. Therefore, the communicating entities should establish a secure channel after authenticating the identity of the communicating entities to ensure their legitimacy as mentioned in [48].

- User Anonymity: as stated in [80], the data originator should be anonymous to internal/external adversaries over the wireless public channel between the resource-

constrained IoT device and the serving edge node. Any Internal/External adversary should not obtain the IoT identity from the authentication request.

- Session Unlinkability: any internal/external adversary cannot acquire the behavior of the low-end device users from their intercepted messages. Therefore, the authentication request from any low-end device should be indistinguishable from a random sequence as mentioned in [80].

- Data Integrity: the sensitive-related data are transmitted over the wireless public channel between the low-end IoT device and the edge node. Therefore, data integrity should be guaranteed against any modification attacks from IoT users to avoid data modification by any active external adversary during its transmission to the edge node [48].

- Perfect Forward Secrecy: this property assures the past session keys are secure even in the case of compromising the long-term key of any communicating party in the system [102]. This is an important property as many experts have raised concerns that individuals and nations around the world are collecting data with the goal in mind of decrypting it at a later date. Data transmitted over the Internet can easily be intercepted and downloaded, and much of that information is sensitive (i.e., Login information, people's credit card numbers, and bank account information are regularly transmitted over the Internet). Perfect Forward Secrecy ensures the security of the data in case of compromising the long-term key.

- Soundness: a synchronization scheme is said to be sound if after a complete run of the protocol, both the communicating parties have updated their internal state at least once to reach a synchronized state and have established the same shared session key [23].

2.1.4 Attacks and Vulnerabilities

In this section, we list the different attacks and highlight their significance in the IoT-edge-cloud paradigm [108].

- Eavesdropping Attack: this attack is launched by an Internal/External attacker to break the confidentiality of the sensitive data transmitted between the communicating entities (IoT devices and edge nodes).

- Active Man-in-the-Middle Attack: this attack occurs when a malicious attacker exists between the IoT device and the edge node and secretly relays or modifies the exchanging data among them. Nevertheless, the IoT device and the edge node believe that they are directly communicating with each other.

- Impersonation Attack: such attack is issued by a malicious adversary to pretend to be a legitimate IoT user to acquire the CO service from the edge node or pretend to be a legitimate edge node to offer fake CO services to IoT users or to launch phishing attack and collect sensitive data from IoT devices.

- Collusion: in such kind of attack two or more entities (i.e., IoT devices or edge nodes) collude to deceive, mislead, or defraud other legal entities.

- Jamming: an attacker intentionally generates a large volume of fake messages to jam communication channels or computing resources, thereby preventing other users from normal communication and computation.

- Data Tampering Attack: this attack is launched by an active adversary to break the integrity of the sensitive information exchanged between the low-end devices and the edge node.

- Hardware Compromise Attack: according to Gartner statistics for the IoT market segment in 2020 [3], a total of 2.38 billion IoTs are expected to be used in unprotected

public sectors, e.g., transportation, retail and wholesale trade, natural resources, and utilities, which makes them an attractive target to physical attacks [35].

- Desynchronization-based Denial of Service Attack (DDSA): such an attack involves an adversary dropping update messages between the IoT device and the edge node during the key agreement phase, preventing future session establishing [132].

- Forgery: malicious attackers might generate false information to mislead other entities. Furthermore, network resources such as bandwidth, storage, and energy would be excessively consumed due to the false data.

2.1.5 Mutual Authentication Protocols for Edge Computing

In what follows, we categorize mutual authentication protocols into Elliptic Curve Cryptography (ECC)-based protocols and lightweight mutual authentication protocols. Moreover, we list authentication protocols for inter-cloud authentication for CO roaming services. ECC-based protocols rely on computationally hard problems. Such protocols incur point multiplication, field exponentiation, or pairing operations which make them suitable only for some applications that do not have severe power and computational constraints. On the other side, efficient and lightweight protocols rely on simple operations such as bitwise XOR operations, hash-based computations, and symmetric encryption which makes them suitable for resource-constrained IoT devices.

ECC-based and Bilinear-based Mutual Authentication Protocols

Hammi *et al.* proposed a one-time password authentication protocol that relies on ECC for smart cities [70]. Additionally, in [90, 95], ECC-based anonymous mutual authentication protocols for energy nodes are proposed for matching the trading energy nodes. Similarly, [91, 158] proposed anonymous mutual authentication protocols for car drivers and park owners. These protocols aim to achieve both the confidentiality of the

transaction and anonymity of the interacting entities in addition to other trading goals like verifiable fairness. Moreover, in [153], to mitigate the long authentication delay and single-point-of-failure in VANETs, Yang *et al.* proposed an edge-assisted decentralized architecture that enables the authentication server to delegate its authentication capabilities to the edge node (Road Side Units). The proposed protocol is public key infrastructure free and it makes use of bilinear pairing to achieve mutual authentication between the fog users and the fog nodes. Azees *et al.* [15] proposed a conditional privacy-preserving authentication scheme where the vehicles and the roadside units generate anonymous certificates on their own, and in the case of dispute, the trusted authority can revoke the real identity of the misbehaving vehicles. Similarly, Jegadeesan *et al.* [79] introduced an anonymous mutual authentication protocol for ensuring data security and user privacy in Wireless Body Area Networks (WBAN). Additionally, Azees *et al.* [16] proposed an affine cipher-based encryption scheme for transferring highly sensitive medical images between patients and doctors. For secure resource management in the smart grid, Chaudhary *et al.* [29] proposed a Software Defined Network (SDN)-based communication model for handling data generation by smart devices. However, the proposed model requires a third party to authenticate the communicating entities and derive the session key. This third party is considered to be a single point of failure and makes the protocol vulnerable to denial-of-service attacks. Unfortunately, the computational and memory requirements for these protocols are beyond the limited capabilities of IoT devices.

Lightweight Mutual Authentication Protocols

Aujla *et al.* [13] proposed an efficient SDN-based protocol for data offloading in the healthcare system. Similar to [29], the main limitation in the proposed protocol is the single point of failure. For the smart grid, a lightweight certificateless mutual authentication protocol has been proposed [119]. Moreover, Li *et al.* [89] proposed an efficient Merkle tree-based authentication protocol for smart meters communications. The protocol en-

sures message integrity and is resilient to replay attacks and message injection attacks. For vehicular networks, Limbasiya *et al.* [97] proposed VCom, a lightweight Vehicle-to-Vehicle communication protocol that is secure to man-in-the-middle, Sybil, and replay attacks. VCom ensures user authentication and anonymity. For a dynamic IoT environment, Dammak *et al.* [37] proposed DLGKM-AC, a decentralized Lightweight Group Key Management for Access Control. The decentralized architecture is composed of one key distribution center and multiple sub-key distribution centers for managing IoT groups and alleviating the rekeying overhead. In the domain of edge computing, Nakkar *et al.* [106] proposed a lightweight one-way authentication protocol for medical emergency systems. The proposed protocol makes use of hash functions and authenticated encryption which makes it post-quantum secure. However, it is a one-way authentication protocol suitable for broadcast messages only. Ibrahim [77] proposed Octopus for lightweight mutual authentication between the edge user and edge server. Octopus suffers from the linkability problem between the messages of the edge user. To thwart adverse consequences of IoT device compromise, PUFs have been considered for electric vehicles [11], smart grids [85], and IoT drones [63] where the edge nodes authenticate the IoT device through a challenge-response query.

Inter-cloud Mutual Authentication Protocols

Some protocols have also been proposed for roaming services. Gope *et al.* [60] proposed a lightweight mutual authentication privacy-preserving protocol with perfect forward secrecy for Global Mobile Networks. The proposed protocol allows a mobile subscriber to get services from a foreign agent. However, the protocol does not consider charging the mobile subscriber and cost redemption for the obtained services. In the context of charging for computation offloading, protocols [39,40,127,146] have been proposed. LAMANCO [146] allows an IoT device to subscribe with a cloud admin for n computation offloading units. Then on serving the IoT device, the IoT gateway authenticates

the IoT device. The massive deployment of LAMANCO is questionable since it requires the deployment of a card reader on IoT devices. Moreover, the interoperability of the protocol between different cloud admins is not achievable as the key materials for authentication are stored in a tamper-proof device in the IoT gateway. Debe *et al.* proposed a transparent decentralized blockchain-based auto-payment protocol [40]. Additionally, an auto-payment protocol that enables the IoT device to select the best computation offloading service providers based on their reputation was proposed in [39] where a smart contract settles the payment procedures between the IoT device and the edge nodes. However, the two protocols do not consider authentication between the IoT device and the edge nodes. Moreover, all the aforementioned protocols do not consider the interoperability between different cloud admins. In [127], the authors proposed an authentication protocol that allows an IoT device to subscribe with the home cloud admin and when moving into other foreign cloud admins, the IoT device provides a verifiable payment token in return for the required computation offloading service. Nevertheless, the protocol does not provide security in the event of a device compromise. More precisely, an attacker can launch a physical attack on the IoT device and obtain the key materials from the compromised device. Later on, the adversary can use these key materials to derive authentication requests from other IoT devices.

2.2 Notation

We use the notation $x \xleftarrow{r} \{0,1\}^n$ to denote the process of uniformly sampling an n-bit value x at random. We also use the notation $Y = Z$ to represent the assignment of the right-hand value Z to the left-hand value Y, while $Y \stackrel{?}{=} Z$ indicates a check of whether the right-hand side Z is equal to the left-hand side Y. The Bitwise exclusive-OR (XOR) is represented by the $\oplus$ symbol. $H(x)$ represents hashing of the string x while $\{M\}_k$ is the AES-CBC encryption of the message M with the key k. Moreover, we used the two

symbols $A||B$ and A, B interchangeably to indicate the concatenation of the two strings A and B together. Throughout the protocols, we use the term IoT gateway to indicate the edge node in the IoT-edge-cloud paradigm.

2.3 Cryptographic Primitives

In what follows, we list the cryptographic primitives that are utilized through our thesis such as PUF, Merkle tree, Schnorr zero-knowledge proof of discrete logarithm knowledge, and Schnorr signatures. We refer to our security parameter as λ. For a function $\epsilon(\lambda) : \mathbb{Z} \rightarrow \mathbb{R}$, we say that it is a negligible function in λ if, for all $d > 0$, there exists $\lambda_d \in \mathbb{Z}$ such that $\epsilon(\lambda) \leq 1/\lambda^d$ for all $\lambda > \lambda_d$ [19].

2.3.1 Physical Unclonable Function

A Physical Unclonable Function is a hardware-based crypto primitive. On stimulating the PUF circuit with an input challenge C, it produces an output response R. PUF can be considered a one-way function that generates a response R for a given input challenge C. In our thesis, we refer to PUF calls as $\mathcal{P}(\ )$. PUFs were first introduced by Gassend *et al.* [54] and can be physically embedded into devices. The output response $R = \mathcal{P}(C)$ is determined by the specific PUF circuit and the challenge input itself. Such challenge-response behavior is unique and determined by the manufacturing process, making it impossible to clone or replicate. Consequently, the challenge-response behavior of the PUF can be used as a biometric identifier for the device. Since most PUFs are noisy, a fuzzy extractor [43] is typically employed to compensate for the variations in the output response. Ideal PUFs are characterized by [122]:

1. *Robustness:* A particular PUF circuit produces the same response R with a very high probability for the same challenge C, even if the experiment is repeated many times.

2. *Uniqueness:* The responses of two different PUFs for the same challenge C are distinguishable with a very high probability.

3. *Unpredictability:* Given the responses of a particular PUF for a set of challenges, it is infeasible to predict the response of this PUF to a new challenge.

4. *Tamper-Evidence:* Any external unauthorized access to the PUF to know the response of the PUF circuit for a certain challenge will alter its physical behavior and accordingly its challenge-response behavior.

5. *Indistinguishability:* The PUF response is computationally indistinguishable from a random sequence of the same length.

Given a PUF $\mathcal{P} : \{0,1\}^{l_1} \rightarrow \{0,1\}^{l_2}$ that maps an l_1-bit input challenge to an l_2-bit output response based on its physical properties, the PUF response should be computationally indistinguishable from a random sequence of the same length. Such a security notion is formally defined as:

Definition 2.1 *A physical unclonable function $\mathcal{P} : \{0,1\}^{l_1} \rightarrow \{0,1\}^{l_2}$ is said to be indistinguishably secure if for any Probabilistic Polynomial Time (PPT) adversary $\mathcal{D}$, $adv_{PUF}^{IND}(\mathcal{D})$ is a negligible function in l_1.*

We can define the adversary advantage against PUF indistinguishability, denoted as $adv_{\mathtt{PUF}}^{IND}(\mathcal{D})$, as the probability of an adversary $\mathcal{D}$ winning a game against a challenger $\mathcal{C}$ that is running the PUF $\mathcal{P}$. The game involves the following steps [143]:

1. During the training phase, $\mathcal{D}$ repeatedly queries the PUF circuit run by $\mathcal{C}$ with challenges C_i and obtains responses $R_i = \mathcal{P}(C_i)$, where $i = 1, 2, \cdots, n$ and n is the number of queries.

2. After the training phase, $\mathcal{D}$ queries the PUF with a challenge C that is different from the ones used in the training phase, $C \notin \{C_1, C_2, \ldots, C_n\}$.

3. The challenger $\mathcal{C}$ generates a random bit $b \xleftarrow{r} \{0,1\}$ and responds with R_b to $\mathcal{D}$, where $R_0 \xleftarrow{r} \{0,1\}^{l_2}$ and $R_1 = \mathcal{P}(C)$.

4. $\mathcal{D}$ outputs its guess $b' \in \{0,1\}$ for b.

The adversary $\mathcal{D}$ wins the game if $b' = b$. The adversary advantage against PUF indistinguishability is defined as $adv_{\mathsf{PUF}}^{IND}(\mathcal{D}) = |\mathsf{Pr}(b' = b) - 1/2|$.

Modeling Attacks and Reconfigurable PUFs

Practical PUFs have been vulnerable to machine learning-based modeling attacks. In such attacks, an adversary, denoted by $\mathcal{D}$, collects a large set of Challenge-Response Pairs (CRPs) $(c_1, r_1), (c_2, r_2), \cdots, (c_m, r_m)$ from the PUF. Using this set, the adversary builds a mathematical model, denoted by $\hat{\mathcal{P}}$ to predict the PUF's response R_{m+1} for a new challenge C_{m+1} [41]. Only strong PUFs are susceptible to modeling attacks, as weak PUFs have a limited set of CRPs, which makes it difficult to train a reliable model using a small amount of data [117].

Reconfigurable PUFs have been proposed as a way to thwart modeling attacks. In particular, dynamic reconfigurable PUFs can change their challenge-response behavior by resetting their configuration and transitioning between states of different challenge-response behavior [138]. Since resetting the PUF configuration renders any previous model useless. An adversary attempting a modeling attack must collect a new set of CRPs to model the new PUF behavior. It is important to note that reconfiguring the PUF does not affect its security properties, such as tamper-evidence and robustness.

DRAM-based PUFs (DPUF) have a very large addressing space and high density for providing a large set of CRPs which makes them suitable for realizing reconfigurable PUFs. In DRAM modules, data is stored in memory cells which are composed of a capacitor and an access transistor. Based on the charge of the capacitor a stored value of 0 or 1 is stored in the memory cell. However, DRAM modules require periodic refreshes of the capacitors to maintain data integrity. In DRAM-based PUFs, the cell power-up and

refresh time are used to generate entropy between the stored challenge and the output response bits of the PUF. A challenge bit string may be stored in an arbitrary memory block of the DRAM module then a refresh-time interval is applied. Then, the stored bit in the memory can be read as the response bit string. For resetting the PUF configuration to realize a new challenge-response behavior, the setting parameters such as the memory block in the DRAM module and the cell refresh time are used. Note that Sutar *et al.* [65] showed that a refresh-time interval of 40-60 ms generates an entropy of $200-500$ bit flips across a 32KB memory block. This new resulting bit-flip behavior is extremely difficult to predict which implies that an adversary with a prior-built model of the PUF has a low probability of predicting the CRPs of its new configuration. Hence, reconfigurable PUFs are secure against modeling attacks. Also, [138] showed that setting a match threshold equal to 27 makes such a design achieve a 100% true-positive rate (i.e., robustness rate) while also ensuring 0% false-positive rate (uniqueness rate).

Fractional Hamming Distance (FHD)

Practical PUFs are known to be noisy, meaning that the output response of a PUF will differ slightly when queried multiple times with the same challenge. To quantify this difference, the Hamming Distance (HD) metric is commonly used. HD measures the difference between two strings by counting the number of substitutions required to change one string into another. Two identical strings will have an HD of zero. FHD is a variant of HD that is normalized by the length of the original string. Specifically, FHD between the strings r and $\hat{r}$ is defined as $\text{FHD}(r, \hat{r}) = \frac{HW(r \oplus \hat{r})}{L(r)}$, where $HW(r \oplus \hat{r})$ is the number of non-zero digits (i.e., 1's) in the XoR of r and $\hat{r}$, and $L(r)$ is the length of the original string r.

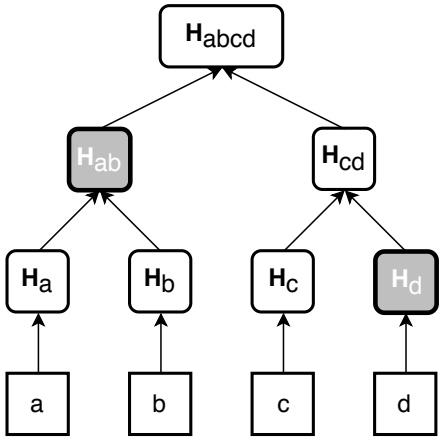

Figure 2.2: An example illustrating the Merkle proof for element $c \in MT$ which consists of the nodes H_d and H_{ab}

2.3.2 Merkle Tree

Merkle trees [103] is a data structure to efficiently and securely verify the integrity and consistency of data. It provides efficient proof of membership for the data. Generally speaking, to accumulate a set of elements, one builds a binary tree where the leaf nodes correspond to the hash values of the elements. The *parent* nodes are assigned the hash of their children using a collision-resistant hash function. The set membership proof, known as *Merkle proof*, has a logarithmic size in terms of the number of leaves. For example, given a Merkle tree MT with a root r, to prove that an element $x \in MT$, the prover sends to the verifier a Merkle proof π which consists of the sibling nodes on the path from x to r as illustrated in Fig. 2.2. The verifier initially computes $r' \leftarrow H(x)$. Then, she iterates sequentially over each hash in π and reconstructs the parent r'. Finally, the verifier accepts the proof π if $r' = r$.

2.3.3 Elliptic Curve

Throughout our work in CP-MAVE and MAPFS, we utilize the ECC system, which is a modern family of public-key cryptosystems based on the algebraic structures of the

elliptic curves over finite fields. ECC security is based on the assumed difficulty of the ECDLP [71, 134]. ECC-based schemes are characterized by smaller key sizes, low arithmetic requirements, and shorter operand lengths compared to RSA systems. Let p be a prime number and let $\mathbb{F}_p$ denote the field of the integers modulo p. An elliptic curve E is defined over the finite field $\mathbb{F}_p$ by the set of the points $(x,\ y) \in\ \mathbb{F}_p \times \mathbb{F}_p$ satisfying non singular elliptic curve equation $y^2 = x^3 + ax + b \bmod p$ such that $4a^3 + 27b^2 \neq 0 \bmod p$ plus the point at infinity $\mathcal{O}$. Then, the additive elliptic curve cyclic group $\mathcal{G}$ is defined as $\mathcal{G} = \{(x, y) : x, y \in\ \mathbb{F}_p \wedge (x, y) \in E\} \bigcup \mathcal{O}$. Let $P \in E$ be the generator point of the group $\mathcal{G}$ of order q. Note that, we use lowercase letters for representing scalar values in Z_q and uppercase letters for EC points in the group $\mathcal{G}$.

The Elliptic Discrete Log Problem (ECDLP) [136]

Definition 2.2 *Elliptic-Curve Discrete Log Problem: Let E be an elliptic curve over the finite field $\mathbb{F}_p$ and let the points R and S points in $\mathcal{G}$ of order q, the ECDLP is the problem of finding an integer r such that $R = rS$.*

The Elliptic Curve Diffie-Hellman (ECDH) Protocol [42]

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

Definition 2.3 *Elliptic-Curve Decisional Diffie-Hellman Problem (ECDDHP) [71]: Let E be an elliptic curve over the finite field $\mathbb{F}_p$. Given the point $P \in \mathcal{G}$ with order q and the points $X = xP$, $Y = yP$, $Z = zP \in \mathcal{G}$, the ECDDHP is the problem of determining whether $Z = xyP$, equivalently, whether $z = xy \bmod q$.*

2.3.4 Schnorr Zero-Knowledge Proof [121]

Given a statement $\mathcal{X}$ where $\exists\ x\ s.t.\ X = xP$, a Schnorr Zero-Knowledge Proof (ZKP) of Knowledge enables the prover to convince the verifier of the knowledge of the witness x without revealing its value to the verifier. Schnorr protocol is a Σ protocol that consists of three interactions between the prover and verifier. These interactions are: commit, challenge, and response. A non-interactive Schnorr ZKP in the Fiat-Shamir heuristic transformation [51] allows the prover to combine the commit, challenge, and response phases in one interaction. This transformation involves using a secure cryptographic hash function to issue the challenge. Moreover, such Fiat-Shamir heuristic transformation is used for generating a Schnorr signature [120] over a message m as follows. The signer generates the commitment $R = rP$ where $r \leftarrow Z_q$ and uses the Fiat-Shamir heuristic transformation for computing the challenge $c = H(R, m)$. Then, it computes the response $s = r + cx$ where the signature $\sigma = (R, s)$. The verifier checks if $sP \stackrel{?}{=} R + cX$ hold.

Chapter 3

Symmetric Key Authentication Protocol with Forward Secrecy

3.1 Introduction

Several authentication protocols have been proposed to enable mutual authentication between communicating entities in the IoT-edge-cloud paradigm, such as those presented in [98, 106, 146, 148]. These protocols achieve various security goals, including mutual authentication between IoT devices and edge nodes, as well as anonymity of the communicating entities and forward secrecy. However, these protocols do not account for the physical vulnerability of the widespread IoT devices, which according to Gartner statistics for the IoT market segment in 2020 [3], are expected to reach 2.38 billion devices deployed in unprotected public sectors, such as transportation, retail and wholesale trade, natural resources, and utilities, making them an attractive target for physical attacks [35].

PUFs have been explored as a way to prevent physical attacks. They have been used in PUF-based protocols for various applications [10,18,92]. Aman *et al.* [10] proposed a PUF-based protocol that is resilient to physical attacks on IoT devices, in the IoT-cloud architecture. However, this protocol is susceptible to Desynchronization-based Denial of

Service Attacks (DDSA). Such attacks involve an adversary dropping update messages between communicating parties during the key agreement phase. Gope *et al.* [62] addressed such an issue by adding a synchronization set of extra pseudo-identities. They also proposed a mutual authentication protocol [64] that uses a reconfigurable PUF to prevent modeling attacks, while mitigating DDSA using extra pseudo-identities stored at the IoT device and the CA. To maintain unlinkability, the requesting IoT device generates a new pseudo-identity with each request. However, with the increasing number of registered IoT devices and the ongoing service request, this impose a pseudo-identity decoding problem and a storage overhead on the CA which scales linearly with the number of IoT devices and the size of the synchronization set.

The existing protocols face various challenges, such as scalability issues with the increasing number of IoT devices, frequent CO service requests, and vulnerability to physical and DDSA. To realize an efficient protocol addressing these challenges, we need to design a mutual authentication protocol for computation offloading that is resistant to physical compromise attacks, and DDSA without the need for storing extra redundant pseudo-identities at the CA. The protocol should have a low-complexity identification algorithm to enhance the efficiency of the CA in decoding IoT pseudo-identities, especially, with the large number of registered IoT devices and frequent IoT requests. Additionally, the proposed protocol has to maintain unlinkability between requests without updating the CA database with each authentication request. Finally, we need to ensure the security of the protocol, which relies on a single long-term key, in case of an IoT device compromise, so that other uncorrupted IoT devices can maintain secure operation.

In this chapter, we propose SKAFS, a **S**ymmetric **K**ey **A**uthentication protocol with **F**orward **S**ecrecy for the IoT-edge-cloud paradigm. The security of the proposed protocol is based on the security of the utilized PUF and hash function. The protocol ensures the anonymity of IoT devices, provides forward secrecy, and is also resistant to hardware compromise attacks, machine learning-based modeling attacks, and DDSA. The

advantages of SKAFS are summarized as follows: (i) our protocol adopts one shared long-term key for decoding the pseudo-identities of the IoT device. However, the protocol still maintains secure operation even with an IoT compromise. This is achieved by keeping the long-term key known only to the CA and the IoT device keeps another private key. Such an IoT private key is then used in the obfuscation operations done by the IoT which are equivalent to obfuscation using the CA's long-term key, (ii) unlike other proposed protocols which, for n IoT devices, require $\mathcal{O}(n)$ search complexity at the cloud server-side for identifying the IoT device from its pseudo-identity, SKAFS requires $\mathcal{O}(1)$ search complexity for identifying the IoT device. This is accomplished by obfuscating the identity of any IoT device with the same long-term key known only to the CA, and (iii) our protocol calls for synchronized states at both IoT devices and cloud admin sides without storing extra redundant pseudo-identities. As such, we embed a synchronization scheme in SKAFS to ensure that the state-update process takes place at the beginning of each session, thus, providing robustness concerning DoS attacks. Our proposed protocol SKAFS preserves the privacy and anonymity of IoT devices and is robust against DDSA, as shown in Table 3.1, which provides a comparison of SKAFS with other existing PUF-based protocols.

Table 3.1: Security/ Functionality properties compared to other IoT-edge-cloud schemes

Security/ Functionality Properties	[10]	[62]	[61]	[92]	[107]	[127]	[123]	[64]	[65]	SKAFS
Mutual Authentication	✓	✓	✓	✓	×	✓	✓	✓	✓	✓
Confidentiality	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Unlinkability	×	✓	✓	×	×	×	✓	✓	✓	✓
Forward Secrecy	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Resilience to DoS	×	✓	✓	✓	×	×	×	✓	✓	✓
Resilience to Physical Attack	✓	✓	✓	✓	✓	×	×	✓	✓	✓
Resilience to Privileged Insider Attack	✓	✓	✓	✓	×	×	✓	✓	✓	✓
Resilience to Modeling Attack	×	×	×	×	×	×	×	✓	✓	✓
Dynamic IoT addition	✓	✓	✓	×	×	×	×	✓	✓	✓
Free of CA updating overhead [1]	×	×	×	NA	NA	✓	✓	×	×	✓
IoT-edge-cloud Compatible	×	×	×	×	✓	✓	✓	×	✓	✓
Identification complexity [2]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	NA	NA	$\mathcal{O}(log(n))$	$\mathcal{O}(log(n))$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$

[1]: There is no need to update the CA with a new IoT pseudo-identity with each authentication request.

[2]: The complexity of the protocol in decoding an incoming IoT pseudo-identity.

✓: The security/functionality feature is achieved.

×: The security/functionality feature is not achieved.

NA: The protocol does not maintain unlinkability through pseudo-identity. Therefore, the functionality feature of updating the CA with a new pseudo-identity and decoding the pseudo-identity is not applicable in that context.

3.2 System Model and Threat model

This section introduces the system model and key entities in our system, followed by the threat model.

3.2.1 System Model

We have adopted the IoT-edge-cloud paradigm, which is comprised of three layers: the IoT layer, the edge layer, and the cloud layer. The cloud layer is operated by the CA, as shown in Fig. 2.1. The edge layer is securely connected to the cloud layer through the core network. To meet the QoS requirements of real-time applications, low-end devices in the IoT layer outsource their heavy computations and storage to the edge server, such as wireless access points and base transceiver stations, in the edge layer via a wireless link. The computational-capable edge server in the proximity of the IoT device performs the computation for the resource-constrained IoT device.

We assume that every IoT device is outfitted with a PUF circuit. Similar to previous works such as [61, 68, 86], we consider the PUF circuit and the IoT microcontroller as a single system on a chip, where the connection between them is regarded as secure. Any tampering with the PUF would change its behavior and render it ineffective. To prevent code modification attacks, it is assumed that the IoT microcontroller is equipped with tamper-proof memory, which guarantees code integrity and protects against malicious code modification attacks through the PUF, as described in [128].

The entities in our proposed protocol are:

1. *Cloud Admin*: The cloud service provider is responsible for deploying the CA, which manages the registration of IoT devices and stores their identities and the corresponding responses from the PUF circuit. To ensure the security of the CA, we assume, as in [65], that the server has access to secure storage and is protected from external interference.

2. *IoT Gateway*: The cloud service provider deploys IoT gateways as access points for IoT devices to access CO services.

3. *IoT Device*: Low-end devices are limited in their resources and have internet accessibility. They are deployed in various applications such as healthcare and virtual reality. To meet the required QoS, IoT devices offload some of their computation and storage to the IoT gateway via a wireless link.

3.2.2 Threat Model

We adopt the Canetti-Krawczyk (CK) adversary model [24], in which the stored information in the memory of any communicating entity is vulnerable to memory leakage attacks. An adversary $\mathcal{A}$ in the CK adversary model can eavesdrop on, drop, insert, or modify messages between the communicating entities, as in the conventional Dolev-Yao threat model [44]. In addition, the adversary has access to stored information such as ephemeral secrets, session keys, and long-term private keys through explicit attacks.

3.3 Protocol Design

In this section, we present the different phases of our proposed protocol. We assume that the registration phase where IoT devices and IoT gateways register with the CA runs in a secure environment. Moreover, throughout the protocol, we assume our keys to be 128 bits. In our protocol, we consider two PUFs: weak PUF $\mathcal{P}_F$ and re-configurable PUF $\mathcal{P}_D$. We consider the weak PUF (e.g., Static Random Access Memory (SRAM)) and non-volatile memory to be embedded in system-on-chip. The weak PUF is used for storing the long-term key on the IoT device. On the other hand, re-configurable PUF $\mathcal{P}_D$ is used in deriving the device's one-time key and it is attached to the device's external memory.

3.3.1 Registration Phase

The CA generates its long term key $K \xleftarrow{r} \{0,1\}^{128}$ and sets its identity to ID_{CA}. The registration phase between IoT_j and CA is illustrated in Fig. 3.1. Note that $j = 1, 2, \cdots, N_{IoT}$ denotes the IoT device index, where N_{IoT} is the total number of registered IoT devices. Initially, IoT_j randomly generates the challenge $C_1 \xleftarrow{r} \{0,1\}^{128}$. Subsequent challenges are computed by applying the hash function over the previous challenge using $C_{i+1} = H(C_i)$ where i is the session index and it starts from 1. On the other hand, C_{F0} and C_{F1} are fixed hardwired challenges for the PUF circuit. Then, the IoT device sends the PUF response for the re-configurable DRAM-based PUF $\mathcal{P}_D^1(C_1)$ and the fixed PUFs $\mathcal{P}_F(C_{F0}), \mathcal{P}_F(C_{F1})$ along with its identity `ID` to the CA. Afterwards, CA computes the bit string $T_j = \mathcal{P}_F(C_{F0}) \oplus \mathcal{P}_F(C_{F1}) \oplus K$ where K is the cloud long-term key, and C_{F0} and C_{F1} are fixed for all IoT devices. The resulting T_j is unique to each IoT device and is stored in its memory. It is different from one device to another based on their PUF responses to C_{F0} and C_{F1}. The CA sends T_j to the IoT device, stores the device's identity `ID`, and initializes the current session key between the IoT device and the IoT gateway, K_c by $\mathcal{P}_D^1(C_1)$. The previous session key, K_p, is initialized with $K_0 \xleftarrow{r} \{0,1\}^{128}$, and the two-step previous K_{p-1} is initialized with $K_{-1} \xleftarrow{r} \{0,1\}^{128}$.

The registration of the IoT gateway l is illustrated in Fig. 3.2 where the IoT gateway sends its identity to the CA. Note that $l = 1, 2, \cdots, N_G$ denotes the IoT gateway index where N_G is the total number of registered IoT gateways. In turn, the CA generates the long-term key MK_{CA-G} and initializes the session key between the CA and the gateway $\mathcal{K}_{CA-G}$ with $H(\mathcal{K}_{CA-G}^p || r_1^p)$ where $\mathcal{K}_{CA-G}^p$ is the previous session key and it is initialized with $\mathcal{K}_0 \xleftarrow{r} \{0,1\}^{128}$ and the random r_1^p is initialized with $r_0 \xleftarrow{r} \{0,1\}^{128}$. Finally, the CA shares both the long-term key and the initial values with the IoT gateway.

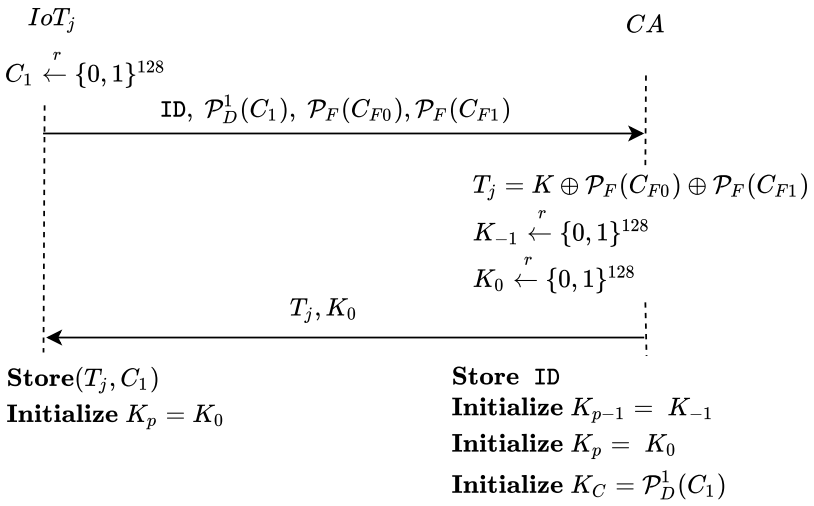

Figure 3.1: IoT device registration

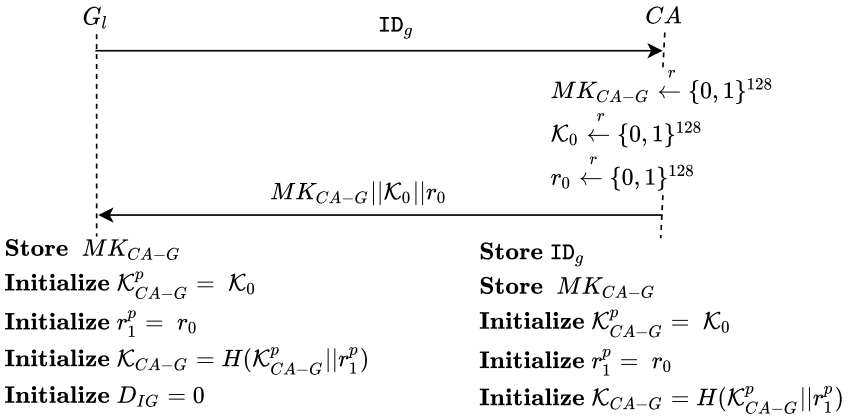

Figure 3.2: IoT gateway registration

3.3.2 Mutual Authentication Phase

Consider the i^{th} session between an IoT device IoT_j and an IoT gateway G_l. Initially, IoT_j sends a Hello message to the G_l. G_l then replies with a random nonce $r_1 \xleftarrow{r} \{0,1\}^{128}$. Afterwards, IoT_j computes its obfuscated identity $\text{ID}^* = H(r_1, r_2) \oplus \text{ID}$, obfuscated key $K_i^* = K_p \oplus K_i$ and the authentication message $M_1 = H(K_i, r_1, r_3)$ where r_2 and r_3 are two randoms chosen by the IoT device $r_2,\ r_3 \xleftarrow{r} \{0,1\}^{128}$. Additionally, IoT_j obfuscates r_2 with the long-term key as shown in Fig. 4.6 where r_2 is XoRed with

the PUF output $\mathcal{P}_F(C_{F0})$ then with the other PUF output $\mathcal{P}_F(C_{F1})$ and finally with T_j. The deletion of both r_2 and the intermediate data (i.e., $\mathcal{P}_F(C_{F0})$, $\mathcal{P}_F(C_{F1})$) thwarts the disclosure of the long-term K in case of a physical attack on IoT$_j$. Following that, IoT$_j$ sends M_1, $\texttt{ID}^*$, r_2^*, r_3^* and K_i^* to the IoT gateway which in turns forwards it to the CA encrypted under the session key between the CA and the IoT gateway, $\mathcal{K}_{CA-G}$. Note that, this shared session key is updated with $\mathcal{K}_{CA-G}^{Next} = H(\mathcal{K}_{CA-G}||r_1)$ for each subsequent session to ensure forward secrecy. Moreover, the IoT gateway sends the messages σ_1, σ_2, $\mathcal{E}_1$ to the CA. Then, the authentication token σ_1 is verified at the CA and σ_2 is used to determine the synchronization difference D_{CA-G} between the IoT gateway and the CA. The CA retrieves the random r_2' and the identity of the IoT device $\texttt{ID}'$. Then, CA picks the stored keys of IoT$_j$ (i.e., two-step previous key K_{p-1}, previous key K_p, and current key K_c). Afterwards, the CA sends σ_2, $\mathcal{E}_2$, D_{CA-G} to the IoT gateway which updates its $\mathcal{K}_{CA-G}$ in case that $D_{CA-G} = -1$ and verifies the authentication token σ_3. Then, to determine the synchronization state of IoT$_j$, the IoT gateway verifies the fractional Hamming distance of K_i and checks M_1 against the current key K_c and the previous key K_p. In case of a successful verification with one of the keys, the IoT gateway sends the authentication message M_2 to IoT$_j$ along with the synchronization difference flag D_{IG}.

3.3.3 Key Agreement Phase

In case of a lagging IoT device where the value of $D_{IG} = -1$, IoT$_j$ updates its challenge $C_i = H(C_i)$ and also sets the configuration of the $\mathcal{P}_D$ to the next state. Then, it computes the synchronized key K_i as illustrated in Fig. 3.4. For the subsequent session, IoT$_j$ sets $\mathcal{P}_D$ to $(i+1)^{th}$ session and computes K_{i+1}. Then, it returns back to the i^{th} session configuration. Afterward, it forwards the obfuscated keys $\hat{K}_{i+1}$ to the IoT gateway which in turn sends it to the CA. The CA updates the synchronization keys K_{p-1}, K_p, K_c, r_1^p, $\mathcal{K}_{CA-G}^P$ and $\mathcal{K}_{CA-G}$. After receiving the confirmation message M_3, the IoT gateway updates the session key $\mathcal{K}_{CA-G}$ and sends the message M_4 to IoT$_j$ to

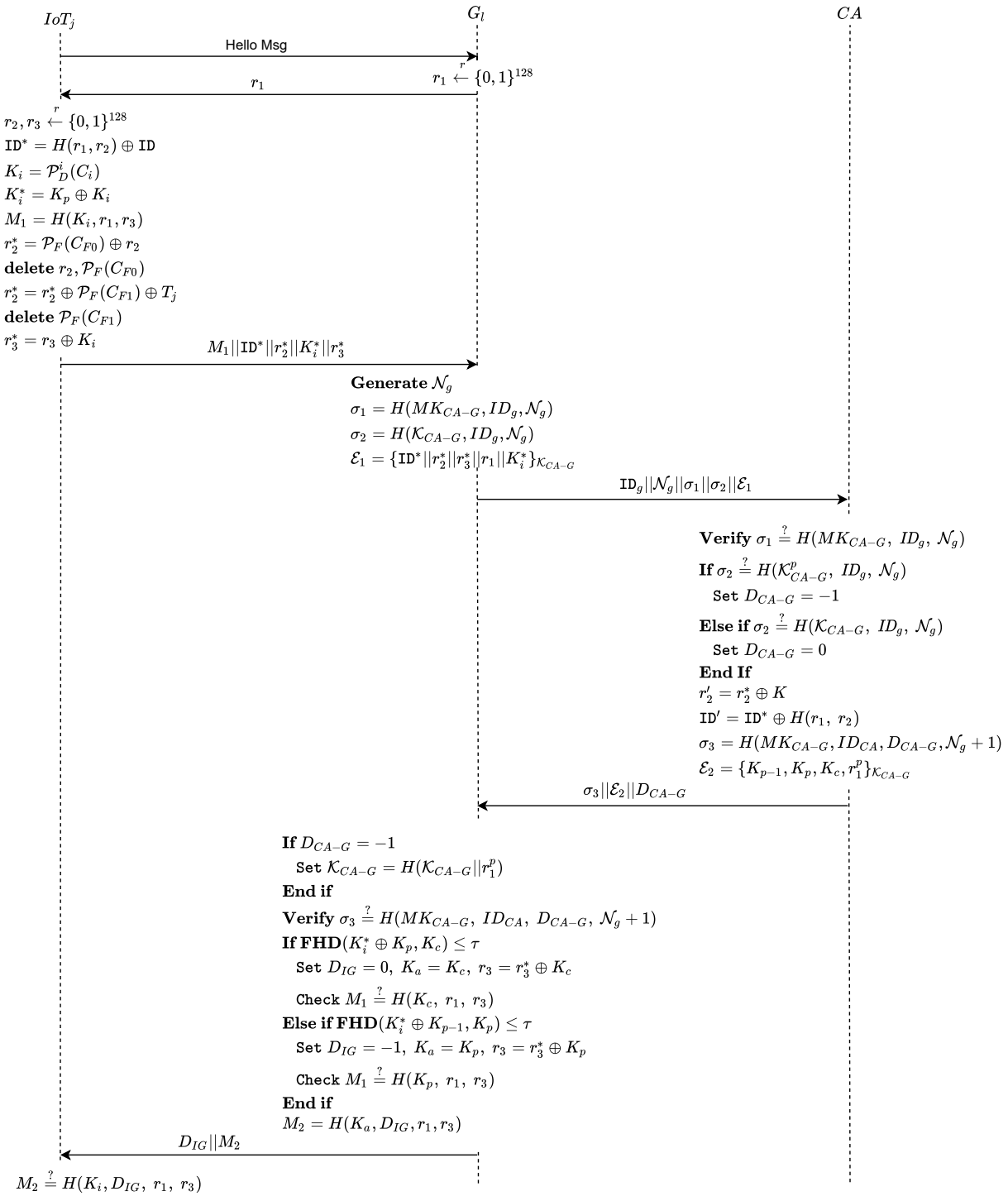

Figure 3.3: Mutual authentication between the IoT device and the IoT gateway for the i^{th} session

confirm the key agreement and synchronization. Finally, IoT_j verifies M_4 and updates its challenge, and reconfigures the $\mathcal{P}_D$ to its new state.

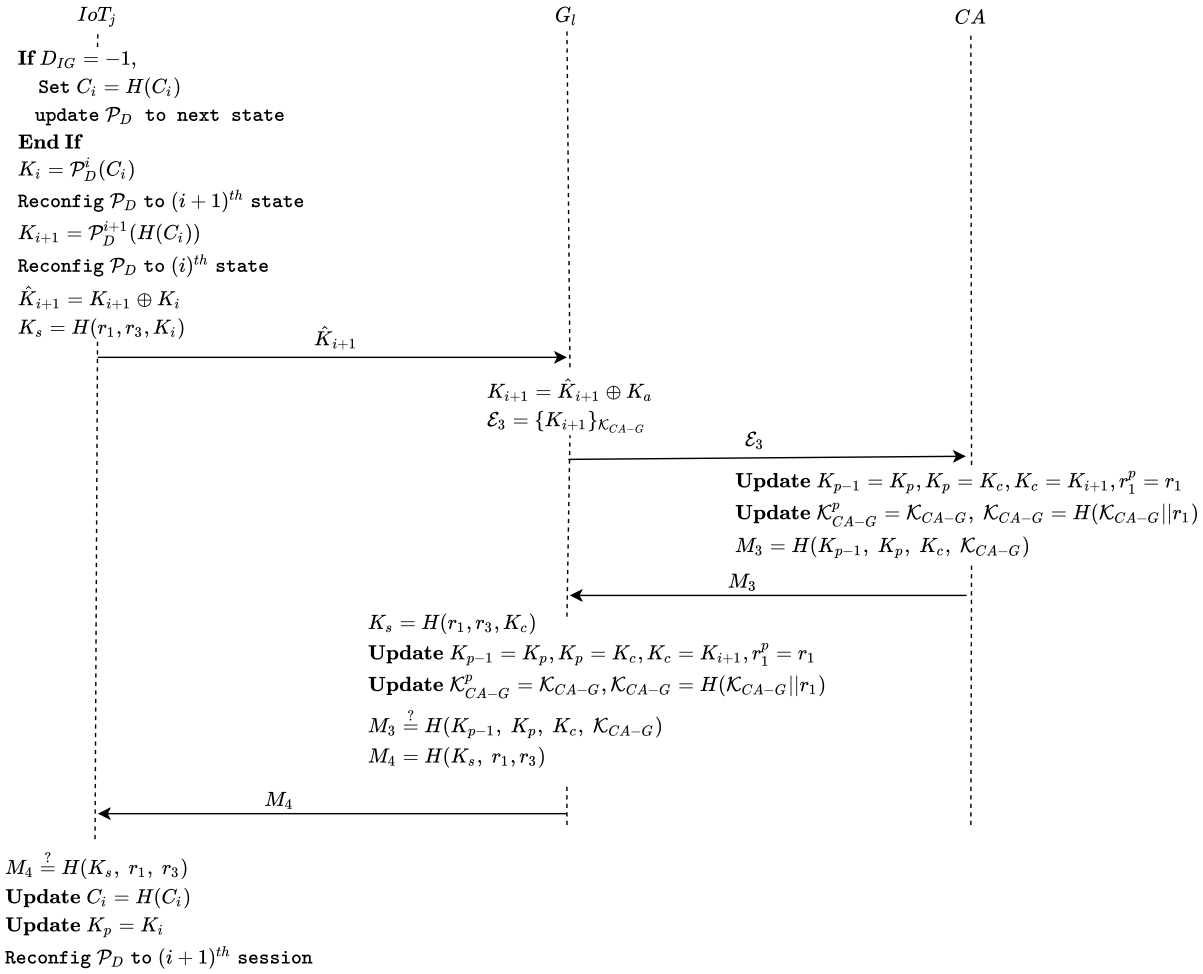

Figure 3.4: Key agreement between the IoT device and the IoT gateway for the i^{th} session

3.3.4 Message Exchange Phase

After mutual authentication and session key agreement, the IoT device starts offloading its heavy computation to the IoT gateway. The IoT device exchanges messages with the IoT gateway encrypted under the session key K_s as shown in Fig. 3.5.

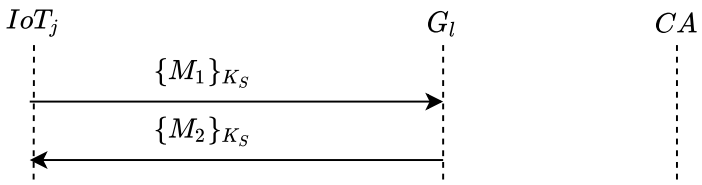

Figure 3.5: Obtaining the CO service offered by the IoT gateway

3.3.5 Dynamic IoT device Addition

The proposed protocol supports the dynamic addition of an IoT device during an up-running instance of the protocol. This can be achieved by the IoT device sending its fixed PUF responses $\mathcal{P}_F(C_{F0})$ and $\mathcal{P}_F(C_{F0})$ and the dynamic PUF response $\mathcal{P}_D^1(C_1)$ to the CA admin. After that, CA responds with the $T = K \oplus \mathcal{P}_F(C_{F0}) \oplus \mathcal{P}_F(C_{F1})$ that the IoT will use later in obfuscating its pseudo-identity.

3.4 Soundness and Formal Security Analysis

In this section, we prove the soundness of the synchronization scheme of SKAFS and provide a formal security analysis of our proposed protocol.

3.4.1 Soundness of SKAFS

According to [14], soundness of a synchronization scheme requires that after a complete correct session between an IoT device and an IoT gateway, the respective internal states of the IoT device, S_{IoT}, and the IoT gateway, S_G, are updated and synchronized. Moreover, the IoT device and the IoT gateway agree on a new session key.

Theorem 3.1 *Let S_{IoT} and S_G denote the IoT device and gateway states, respectively. At the i-th session, let K_c^{IoT} and K_n^{IoT} denote the current and next session keys at the IoT device, respectively. Similarly, let K_p^G and K_c^G denote the previous and current session keys at the IoT gateway. SKAFS ensures that the following two conditions always hold.*

1. *Either* $(S_{IoT},\ S_G) = (S_i,\ S_i)$ *indicating synchronized states, or* $(S_{IoT},\ S_G) = (S_{i-1},\ S_i)$ *indicating a lagging IoT device state, can exist.*

2. *After the completion of a correct session, both the IoT device and the gateway must have updated their internal states at least once, and* $K_c^{IoT} = K_c^G$.

Proof: Considering the first case where the IoT device and the IoT gateway are synchronized (i.e., $D_{IG} = 0$), the IoT gateway verifies the authentication message M_1 using K_c^G. Then, the gateway sets $D_{IG} = 0$ and sends the authentication message M_2 to the IoT device for computing the session key. Therefore, starting with $D_{IG} = 0$ and after receiving each of $M_1,\ M_2,\ K_{i+1}$ and M_4, one can obtain the following IoT and gateway states S_{IoT} and S_G for the session i

1. Initially,
   $(S_{IoT},\ S_G) = (S_i,\ S_i),\ D_{IG} = 0.$

2. After receiving M_1 by the IoT gateway,
   $(S_{IoT},\ S_G) = (S_i,\ S_i),\ D_{IG} = 0.$

3. After receiving M_2 by the IoT device,
   $(S_{IoT},\ S_G) = (S_i,\ S_i),\ D_{IG} = 0.$

4. After receiving K_{i+1} by the IoT gateway,
   $(S_{IoT},\ S_G) = (S_i,\ S_{i+1}),\ D_{IG} = -1.$

5. After receiving M_4 by the IoT device,
   $(S_{IoT},\ S_G) = (S_{i+1},\ S_{i+1}),\ D_{IG} = 0.$

The evolution of the respective internal states in the IoT device and the IoT gateway are listed in Table 3.2. We can see that upon receiving M_2, the IoT device and the gateway are synchronized (i.e., $D_{IG} = 0$) and they agree on the same session key (i.e., $K_c^{IoT} = K_i$ and $K_c^G = K_i$). Moreover, after a complete run of SKAFS, both the IoT device and the

gateway update their internal states at least once (i.e., $(S_{IoT},\ S_G) = (S_{i+1},\ S_{i+1})$). Note that the possible resulting D_{IG} values are either 0 or -1, and the case where $D_{IG} = 1$ does not occur since the IoT device updates its state only upon receiving the confirmation message M_4.

Investigating the other case where the IoT device is lagging, i.e., $D_{IG} = -1$, the IoT gateway has K_p^G, and K_c^G from the previous session, the IoT gateway verifies the authentication message M_1 using its K_p^G. Then, the IoT gateway sets $D_{IG} = -1$ and sends the authentication message M_2 to the IoT device for synchronization and session key computation. After M_2 verification, the IoT device updates its internal state and computes the session key. Therefore, starting with $D_{IG} = -1$, after the authentication messages $M_1,\ M_2,\ K_{i+1}$ and M_4, one can obtain the following IoT and gateway states S_{IoT} and S_G for the session i

1. Initially,

   $(S_{IoT},\ S_G) = (S_{i-1},\ S_i)$, $D_{IG} = -1$.

2. After receiving M_1 by the IoT gateway,

   $(S_{IoT},\ S_G) = (S_{i-1},\ S_i)$, $D_{IG} = -1$.

3. After receiving M_2 by the IoT device,

   $(S_{IoT},\ S_G) = (S_i,\ S_i)$, $D_{IG} = 0$.

4. After receiving K_{i+1} by the IoT gateway,

   $(S_{IoT},\ S_G) = (S_i,\ S_{i+1})$, $D_{IG} = -1$.

5. After receiving M_4 by the IoT device,

   $(S_{IoT},\ S_G) = (S_{i+1},\ S_{i+1})$, $D_{IG} = 0$.

The evolution of the IoT device and the IoT gateway internal states are listed in Table 3.3. We can see that after the reception and verification of the authentication message M_2, the IoT device updates its internal state and becomes synchronized with the

IoT gateway (i.e., $D_{IG} = 0$); meanwhile, the IoT device and the IoT gateway agree on the same session key (i.e., $K_c^{IoT} = K_i$ and $K_c^G = K_i$). Moreover, during a complete run of SKAFS, the IoT device updates its internal states twice while the IoT gateway updates its internal state once (i.e., initially, $(S_{IoT},\ S_G) = (S_{i-1},\ S_i)$ and at the end $(S_{IoT},\ S_G) = (S_{i+1},\ S_{i+1})$). Note that the possible resulting D_{IG} values in the lagging case $\in \{0, -1\}$. ■

Table 3.2: The evolution of the internal states in case $D_{IG} = 0$

Epoch	state	D_{IG}	K_c^{IoT}	K_n^{IoT}	K_p^G	K_c^G
Initial	(S_i, S_i)	0	K_i	K_{i+1}	K_{i-1}	K_i
After M_1	(S_i, S_i)	0	K_i	K_{i+1}	K_{i-1}	K_i
After M_2	(S_i, S_i)	0	K_i	K_{i+1}	K_{i-1}	K_i
After $\hat{K}_{i+1}$	(S_i, S_{i+1})	-1	K_i	K_{i+1}	K_i	K_{i+1}
After M_4	(S_{i+1}, S_{i+1})	0	K_{i+1}	K_{i+2}	K_i	K_{i+1}

Table 3.3: The evolution of the internal states in case D_{IG} = -1

Epoch	state	D_{IG}	K_c^{IoT}	K_n^{IoT}	K_p^G	K_c^G
Initial	(S_{i-1}, S_i)	-1	K_{i-1}	K_i	K_{i-1}	K_i
After M_1	(S_{i-1}, S_i)	-1	K_{i-1}	K_i	K_{i-1}	K_i
After M_2	(S_i, S_i)	0	K_i	K_{i+1}	K_{i-1}	K_i
After $\hat{K}_{i+1}$	(S_i, S_{i+1})	-1	K_i	K_{i+1}	K_i	K_{i+1}
After M_4	(S_{i+1}, S_{i+1})	0	K_{i+1}	K_{i+2}	K_i	K_{i+1}

Using the same approach as in Theorem 3.1, we can prove that the states of the IoT gateway and the CA in the link between the IoT gateway and the CA are updated and synchronized after a complete run of the protocol and the two communicating entities agree on a new session key $\mathcal{K}_{CA-G}$.

3.4.2 Security Analysis of SKAFS using AVISPA

In what follows, we provide our security analysis of the proposed protocol using SPAN+AVISPA (Security Protocol ANimator for Automated Validation of Internet Security Protocols and Applications) framework [12]. AVISPA is used for modeling and

formally analyzing security protocols. AVISPA ensures that the security protocol is free from active and passive attacks. Furthermore, it determines whether the protocol is vulnerable to replay and Man-in-the-Middle security attacks.

High-Level Protocol Specification Language (HLPSL) is used to describe security protocols in AVISPA. Afterward, the HLPSL script is translated into an Intermediate Format (IF) specification using the HLPSL2IF translator. Such an IF is analyzed by back-end tools to generate the Output Format (OF). AVISPA comprises four back-end tools namely: On-the-fly Model-Checker (OFMC), Constraint Logic based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC), and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP). Only OFMC and CL-AtSe tools support exclusive-OR operations.

HLPSL describes each protocol entity in a module called a basic role which specifies what information the entity possesses, its initial state, and the transition between the states. An intruder is modeled using the channel (dy) which stands for the Dolev-Yao intruder model [44].

In our analysis of SKAFS we used the CL-AtSe back-end tool as it supports the exclusive-OR operations. The generated OF shown in Fig. 3.6 illustrates that the protocol is safe under the CL-AtSe back-end. This means that SKAFS is secure against Man-in-the-middle attacks and satisfies the secrecy of the session key, the secrecy of IoT ID, and the mutual authentication between the IoT device, IoT gateway, and the CA as specified in the environment role.

3.4.3 Formal Security Analysis

In this section, we start by modeling the adversarial capabilities. Then, we provide the formal security proofs for our protocol. For simplicity, we proceed with our proof by assuming that a practical PUF on the IoT device followed by fractional hamming distance measurement on the IoT gateway side can be modeled as an ideal PUF as in [64, 65].

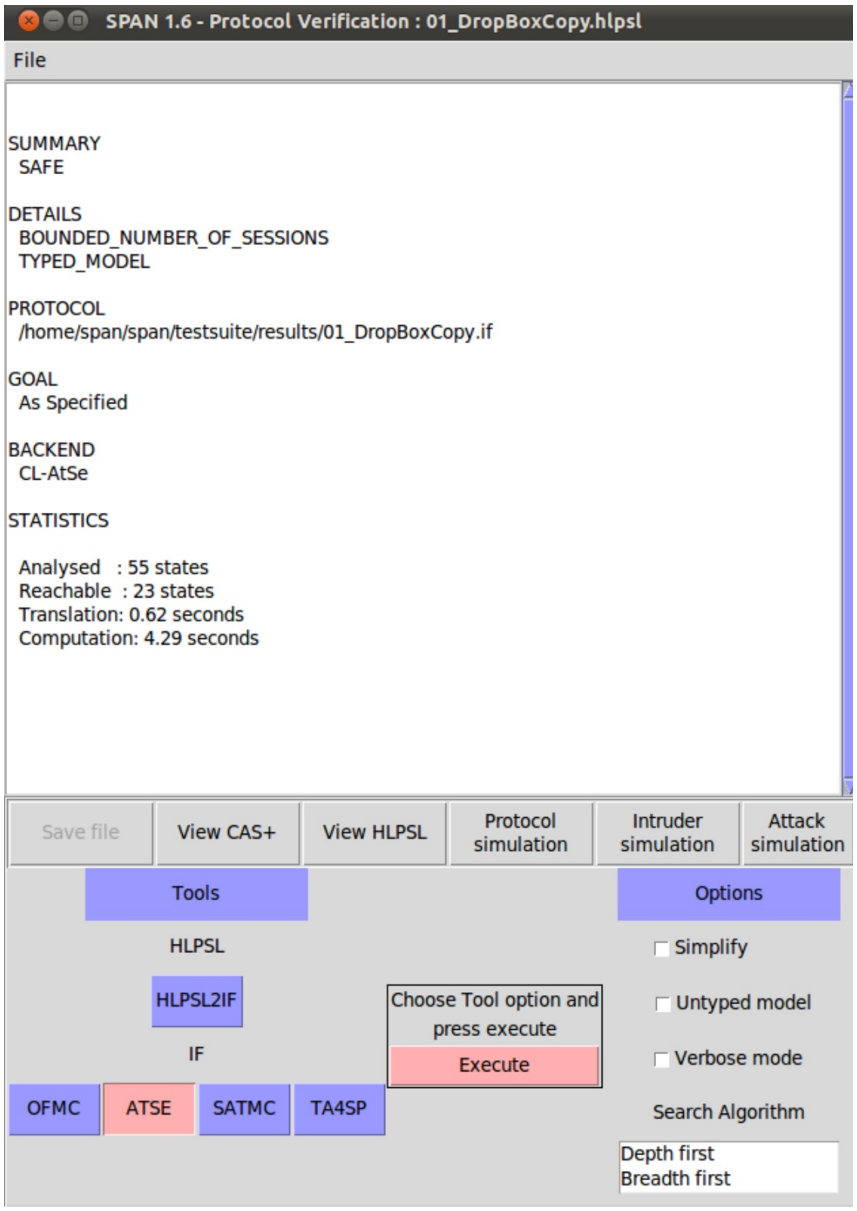

Figure 3.6: AVISPA results for the specified goals (i.e., the secrecy of the session key, the secrecy of IoT ID, and authentication of the IoT device, the IoT gateway, and the CA)

Adversarial Oracles

Under the CK-threat model discussed in Section 3.2.2, an adversary $\mathcal{A}$ has control over the wireless communication channel between the IoT device and the IoT gateway. Furthermore, $\mathcal{A}$ has access to several oracles:

- **Launch**(1^{λ})**:** Creates a fresh protocol instance with identifier π.

- **Send IoT**(m, IoT)**:** Models the ability of the adversary to act as a legitimate IoT gateway. In this context, $\mathcal{A}$ sends a message m to IoT in the protocol instance π, and IoT responds with message m'.

- **Send Gateway**(m, G) **:** Models the adversary's capacity to act as a legitimate IoT device. In this context, $\mathcal{A}$ sends message m to the IoT gateway G in the protocol instance π, and G responds with message m'.

- **Execute**(IoT, G)**:** Models the adversary's ability to observe the radio communication channel and intercept a protocol instance π between an IoT device and an IoT gateway. It generates the transcript $\mathcal{T}$ of the transmitted messages of the protocol instance π during its execution between IoT and G.

- **SSReveal**(U)**:** Allows the adversary $\mathcal{A}$ to learn the session state information inside the entity U.

- **SKReveal**(U)**:** Allows the adversary $\mathcal{A}$ to learn the session key of the entity U.

- **Corrupt**(U)**:** Allows $\mathcal{A}$ to learn the long-term private key of the entity U. It is worth noting that according to [55], a physical attack on memory is always invasive and ultimately destroys its on-chip neighboring PUF. Therefore, the **SSReveal, SKReveal**, and **Corrupt** oracles can be invoked only once on an IoT device. $\mathcal{A}$ cannot access the PUF output afterward and the IoT is rendered useless. This assumption is consistent with [61].

In what follows, we proceed with our security proofs for mutual authentication, IoT device anonymity and hardware compromise resilience.

Theorem 3.2 *Our proposed SKAFS protocol achieves mutual authentication between the IoT device and the IoT gateway.*

Proof: An adversary $\mathcal{A}$ may want to impersonate a legitimate IoT device or a legitimate IoT gateway. This can be modeled by the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. In the training phase, the challenger $\mathcal{C}$ chooses a legitimate IoT device IoT and gateway G.
2. $\mathcal{A}$ interacts with SKAFS (i.e., IoT and G) using the adversary oracles.
3. After the training phase, $\mathcal{A}$ notifies the challenger $\mathcal{C}$.
4. $\mathcal{A}$ calls **Send Gateway** oracle to impersonate IoT.

$\mathcal{A}$ wins the game if it can send a valid authentication message M_1 to impersonate a legitimate IoT device IoT to the IoT gateway G. The adversarial advantage against the mutual authentication is denoted by $adv^{auth}_{SKAFS}(\mathcal{A})$.

Assuming a PPT adversary $\mathcal{A}$ who can break the mutual authentication of SKAFS, this adversary can function as a subroutine in a deterministic algorithm of an adversary $\mathcal{D}$ to break the indistinguishability property of the PUF. For impersonating an IoT, $\mathcal{D}$ intercepts r_1, randomly generates r_2 and uses $\mathcal{A}$ to invoke the **SSReveal** oracle on IoT to know the current challenge C_i. $\mathcal{D}$ passes r_1 and r_2 to $\mathcal{A}$ where $\mathcal{A}$ generates the authentication message M_1. Then, $\mathcal{D}$ challenges PUF (i.e., in the PUF indistinguishability game) with C_i and gets R_b as the response. $\mathcal{D}$ outputs $b = 1$ when $H(R_b, r_1, r_2) = M_1$ and $b = 0$ otherwise (i.e., $adv^{IND}_{PUF}(\mathcal{D}) = adv^{auth}_{SKAFS}(\mathcal{A})$). Since we assume an ideal and secure PUF, therefore, the advantage of $\mathcal{A}$ in forging M_1 message and impersonating an IoT device is $adv^{auth}_{SKAFS}(\mathcal{A}) = adv^{IND}_{PUF}(\mathcal{D}) \leq \epsilon$.

For impersonating an IoT gateway to the IoT device, $\mathcal{D}$ intercepts r_1 and D_{IG} and uses $\mathcal{A}$ to invoke **SSReveal** oracle on IoT to know the current challenge C_i and r_3. $\mathcal{D}$ passes r_1 and D_{IG} to $\mathcal{A}$ where $\mathcal{A}$ generates the authentication message M_2. Then, $\mathcal{D}$ challenges the PUF (i.e., in the PUF indistinguishability game) with C_i and

gets R_b as the response. $\mathcal{D}$ outputs $b = 1$ when $H(R_b, D_{IG}, r_1, r_3) = M_2$ and $b = 0$ otherwise (i.e., $adv^{IND}_{PUF}(\mathcal{D}) = adv^{auth}_{SKAFS}(\mathcal{A})$). Since we assume an ideal and secure PUF, the advantage of $\mathcal{A}$ in forging M_2 message and impersonating an IoT gateway is $adv^{auth}_{SKAFS}(\mathcal{A}) = adv^{IND}_{PUF}(\mathcal{D}) \leq \epsilon$. Therefore, SKAFS achieves mutual authentication. ■

Theorem 3.3 *Our proposed SKAFS protocol is privacy-preserving.*

Proof: The anonymity of the sender means that an attacker cannot sufficiently identify the sender within the sender's anonymity set, while unlinkability of two or more items of interest (e.g., messages, and actions) means that an attacker cannot sufficiently distinguish if these two items are related or not [112]. Unlinkability is a sufficient condition of anonymity [72]. Therefore, when SKAFS achieves unlinkability between the sessions, it also achieves the anonymity of the IoT device.

SKAFS is privacy-preserving (i.e., unlinkable) if a PPT adversary $\mathcal{A}$ cannot correlate two successful authentication requests and responses of an IoT device with a gateway. This can be modeled by the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. In the training phase, the challenger $\mathcal{C}$ chooses a gateway G and two legitimate IoT devices IoT_0 and IoT_1.
2. $\mathcal{A}$ interacts with SKAFS (i.e., IoT_0, IoT_1 and G) using the adversary oracles.
3. After the training phase, $\mathcal{A}$ notifies $\mathcal{C}$.
4. $\mathcal{C}$ samples a bit $b \xleftarrow{r} \{0, 1\}$ and selects IoT device IoT_b
5. $\mathcal{A}$ interacts with IoT_b and G.
6. $\mathcal{A}$ outputs her guess $b' \in \{0, 1\}$ for b. $\mathcal{A}$ wins the game if $b' = b$.

The advantage of $\mathcal{A}$ against the anonymity of SKAFS is defined as $adv^{unlink}_{SKAFS}(\mathcal{A}) = |\mathsf{Pr}(b' = b) - 1/2|$. SKAFS is privacy-preserving if a PPT $\mathcal{A}$ cannot distinguish between the obfuscated identities of two IoT devices (i.e., the adversary advantage $adv^{unlink}_{SKAFS}(\mathcal{A}) \leq \epsilon$).

Let us assume a PPT adversary $\mathcal{A}$ who breaks the anonymity of SKAFS and can distinguish between the obfuscated identities of two IoT devices. This implies that $\mathcal{A}$ can distinguish between obfuscated identities of an IoT device and a random sequence. In what follows, we show that $\mathcal{A}$ can be used by adversary $\mathcal{D}$ to break the PUF indistinguishability property. $\mathcal{D}$ uses $\mathcal{A}$ to invoke **SSReveal** and **Corrupt** oracles on an IoT device with identity ID to get the fixed challenges C_{F0}, C_{F1} and T. Then, in the PUF indistinguishability game, in the training phase, $\mathcal{D}$ sends C_{F0} to get $\mathcal{P}(C_{F0})$ and in the challenge phase, $\mathcal{D}$ sends C_{F1} and get R_b. $\mathcal{D}$ passes r_1, r_2^*, and $\mathtt{ID}^* = H(r_1, r_2^* \oplus T \oplus \mathcal{P}(C_{F0}) \oplus R_b) \oplus \mathtt{ID}$ to $\mathcal{A}$. If $\mathtt{ID}^*$ is a valid obfuscated identity, $\mathcal{A}$ outputs $b = 1$ as it can distinguish the obfuscated identity of an IoT device with identity ID from a random sequence. Otherwise $\mathcal{A}$ outputs $b = 0$. In the PUF indistinguishability game, $\mathcal{D}$ outputs $\mathcal{A}$'s guess b (i.e., $adv_{PUF}^{IND}(\mathcal{D}) = adv_{SKAFS}^{unlink}(\mathcal{A})$). Since we assume an ideal and secure PUF, then $adv_{SKAFS}^{unlink}(\mathcal{A}) = adv_{PUF}^{IND}(\mathcal{D}) \leq \epsilon$, therefore, SKAFS is privacy-preserving. ■

Theorem 3.4 *Launching a physical attack on the IoT device and having access to the stored information does not leak any additional information regarding the CA long-term Key K.*

Proof: Hereby, we prove that the long-term key K does not reside on the IoT device memory at any time. An adversary $\mathcal{A}$ with access to the **SSReveal, SKReveal, Corrupt** oracles does not get any additional information on the long-term key K more than what it gets by performing a passive attack on the communication channel (i.e., invoking the **Execute** oracle). Specifically, an adversary can compromise the IoT device before the $\mathcal{P}_F(CF_0)$ to get the random r_2 or after the $\mathcal{P}_F(CF_0)$ to get r_2 and $r_2 \oplus \mathcal{P}_F(C_{F0})$ or after the 2^{nd} PUF call $\mathcal{P}_F(C_{F1})$ to get $r_2 \oplus \mathcal{P}_F(C_{F0}) \oplus \mathcal{P}_F(C_{F1})$. Subsequently, compromising an IoT device enables $\mathcal{A}$ to gain access to one of these sets.

1. r_2, r_3, ID, T_j
2. r_2, r_3, ID, ID*, T_j

3. r_2, r_3, ID, ID*, K$_i$, T_j

4. r_2, r_3, ID, ID*, K$_i$, K$_i^*$, T_j

5. r_2, r_3, ID, ID*, K$_i$, K$_i^*$, $r_2 \oplus \mathcal{P}_F(C_{F0})$, T_j

6. r_3, ID, ID*, K$_i$, K$_i^*$, $r_2 \oplus \mathcal{P}_F(C_{F0})$, $r_2 \oplus K$, T_j

7. r_3, r_3^*, ID, ID*, K$_i$, K$_i^*$, $r_2 \oplus K$, T_j

Since the cloud long-term key $K = \mathcal{P}_F(C_{F0}) \oplus \mathcal{P}_F(C_{F1}) \oplus T_i$, getting r_2, r_3, ID, T_j leaks no information on the cloud key K. Getting r_2, r_3, ID, ID*, T_j is equivalent to getting r_2, r_3, ID, ID*, T_j , $H(r_1, r_2)$. Under the assumption of a preimage-resistant hash function, the additional information is of no value to the attacker. Getting r_2, r_3, ID, ID*, K$_i$, T_j is equivalent to getting r_2, r_3, ID, ID*, K$_i$, T_j, $H(r_1, r_2)$. Similar to the previous case, the additional information is of no value to the attacker. Getting r_2, r_3, ID, ID*, K$_i$, K$_i^*$, T_j (resp. r_2, r_3, ID, ID*, K$_i$, K$_i^*$, $r_2 \oplus \mathcal{P}_F(C_{F0})$, T_j) is equivalent to getting r_2, r_3, ID, ID*, K$_i$, K$_i^*$, T_j, K_p (resp. r_2, r_3, ID, ID*, K$_i$, K$_i^*$, $\mathcal{P}_F(C_{F0})$, T_j). Assume the length of the PUF response to be l_r and $\mathcal{P}_F(C_{F1})$ to be a random value of entropy l_r. Since $K = \mathcal{P}_F(C_{F0}) \oplus \mathcal{P}_F(C_{F1}) \oplus T_i$, therefore, the cloud long-term key K is still a random value of entropy l_r. Similarly, getting r_3, ID, ID*, K$_i$, K$_i^*$, $r_2 \oplus \mathcal{P}_F(C_{F0})$, $r_2 \oplus K$, T_j is equivalent to getting r_3, ID, ID*, K$_i$, K$_i^*$, $r_2 \oplus \mathcal{P}_F(C_{F0})$, $r_2 \oplus K$, $\mathcal{P}_F(C_{F1})$, T_j. Hence, K remains a random value with entropy l_r. Getting r_3, r_3^*, ID, ID*, K$_i$, K$_i^*$, $r_2 \oplus K$, T_j does not leak information on the long-term key K. ■

Perfect Forward Secrecy

A protocol achieves perfect forward secrecy if compromising the long-term key or the current session key does not lead to the disclosure of the past session keys. In our protocol, the session key for the session i is computed as $K_S = H(r_1, r_3, K_i)$. The IoT gateway generates a random number r_1 and sends it to the IoT device in the clear. The

IoT device generates a random number r_3 and obfuscates it using the instantaneous key K_i, which is randomly generated by the PUF circuit and obfuscated with the previous-session PUF key K_{i-1}. Therefore, this protocol ensures perfect forward secrecy because the past session keys are independent of the long-term key and cannot be derived from a compromised session key.

Backward Secrecy

A protocol is said to achieve the backward secrecy property if an adversary who has access to the protocol state values of the current session, cannot calculate the past session keys. We have listed in Theorem 3.4 all of the state information of the IoT device during the authentication phase. Since the fresh session key depends on randoms freshly generated by the IoT gateway, the IoT device, and the embedded PUF $K_s = H(r_1, r_2, K_i)$, the previous session keys are still secured and the protocol achieves the backward secrecy property.

Replay Attacks

After receiving a hello message from the IoT device, the IoT gateway generates the random r_1 and sends it to the IoT device. Subsequently, the IoT device generates the randoms r_2 and r_3. Afterwards, authentication messages M_1, M_2 and M_4 incurs the randoms r_1, r_3. Therefore, any replay attack of a previous authentication request of $r_1^{'}$ and $r_3^{'}$ values will not be accepted by the IoT device with fresh r_3 nor by the IoT gateway with fresh r_1.

Attacks Exploiting the IoT Gateway

1. *Privilege Insider Attack.* According to [115], an insider attack is a malicious attack caused by an entity inside the organization. In our case, we consider the privileged insider attack originating from the IoT gateway. We exclude the CA and consider it

as a trusted authority like in the IoT-Centric Cloud paradigm [21, 137]. If the IoT gateway can deviate from the design goals during the mutual authentication with the IoT device and gets access to extra information, we would say that SKAFS is prone to privileged insider attack.

The requesting IoT sends M_1, $\texttt{ID}^*$, r_2^*, r_3^* and K_i^* to the IoT gateway which in turns forwards it to the CA. Note that, $\texttt{ID}^*$ is obfuscated with $\mathrm{H}(r_1,\ r_2)$, r_2^* is obfuscated with the long-term key K, r_3^* is obfuscated with the session key K_i and the current session key on the IoT device K_i^* is obfuscated with the previous session key K_p. The cloud admin identifies the requesting IoT device and responds with two-step previous key $K_{p-1} = K_{i-2}$, previous Key $K_p = K_{i-1}$, and current key $K_c = K_i$ to the IoT gateway for synchronization and session key establishment. Therefore, with this additional information, the IoT gateway gets access to the current session key and r_3 without any extra information on r_2 or the identity of the requesting IoT. Moreover, since the IoT gateway does not have access to the identity of the requesting IoT or the long-term key K, it cannot launch an impersonation attack on other IoT devices to other IoT gateways. Therefore, our proposed protocol is immune to privileged insider attacks.

2. *Stolen Verifier Attack.* According to [31], a stolen verifier attack is where an adversary who has stolen the verifier token for the user password can impersonate that user to the cloud server later. By employing a dictionary attack, the stolen verifier attack can be achieved on low-entropy passwords. Here, in our protocol, the IoT gateway does not store verifier data. Instead, it obtains it from the CA admin after receiving the IoT authentication request. Moreover, the IoT device uses the DPUF response for generating the authentication keys which have more entropy and are updated after each session. Thus, stealing the verifier token from the IoT gateway is not possible unless for the running session only. Even if the adversary gets access to the current verifier, this verifier will be useless for the next session which requires

an updated authentication key.

3.5 Performance Analysis and Comparison

Generally, IoT devices are limited in their storage and computation resources. Thus, it is important to consider the efficiency of SKAFS by analyzing its storage and computation requirements. In our evaluation, the storage overhead is indicated by the size of the memory required by the IoT device and the CA to store the secrets needed during the mutual authentication and key establishment phases. The computation requirement is measured by the cryptographic operations on the IoT device and the IoT gateway to achieve mutual authentication and establish the session key before the actual exchange of the data. The communication overhead between the IoT device and the IoT gateway is the messages exchanged between them before the actual exchange of the data. The performance evaluation results are summarized in Table 3.4.

Table 3.4: Storage, Computation and Communication Cost

Description	Value
IoT storage	64 bytes
Cloud storage	$N_{IoT} \times 4 \times 16 + N_G \times 2 \times 16$ bytes
IoT computation	2 RNG, 7 hash, 2 $\mathcal{P}_D(\ )$, 2 $\mathcal{P}_F(\ )$
Gateway computation	2 RNG, 7 hash, 2 Enc, 1 Dec, 2 FHD
CA computation	4 hash, 1 Enc, 2 Dec
Communication IoT-G	192 bytes
Communication G-CA	320 bytes

3.5.1 Communication Cost

Throughout this section, we consider SHA-256 and AES-CBC encryption mode as in [58]. For the identity of the IoT device, we assume a 128-bit identity and 1-bit D_{IG}. For the PUF circuit parameters, we assume a challenge of 128 bits as stated in [10] and a PUF response of 128 bits, as stated in [58].

In the mutual authentication phase, the IoT gateway sends r_1 which is a 128-bit message and the IoT device replies with a 6×128 bits message of $\{M_1,\ \texttt{ID}^*,\ r_2^*,\ K_i^*,\ r_3^*\}$. Then, the IoT gateway sends a 1-bit D_{IG} and the authentication message M_2 of 256 bits. Later, in the synchronization phase, the IoT device sends the next session key K_{i+1} of 128 bits, and the IoT gateway confirms with a 256-bit M_4. Thus, in total, the communication cost between the IoT device and the IoT gateway is 192 bytes plus 1-bit D_{IG}. On the other side, the communication cost between the IoT gateway and CA is 320 bytes.

3.5.2 Storage Requirements

For each IoT device, the server needs storage of 4×128 bits for keeping the IoT's identity ID, K_{p-1}, K_p, and K_c. Also, the server requires 2×128 bits for storing the master key and the session key with each deployed IoT gateway. On the other side, the IoT device stores its identity ID, the bit string T, the session's challenge C_i, and the previous key K_p. Thus, in total, the IoT device requires a total storage of 4×128 bits.

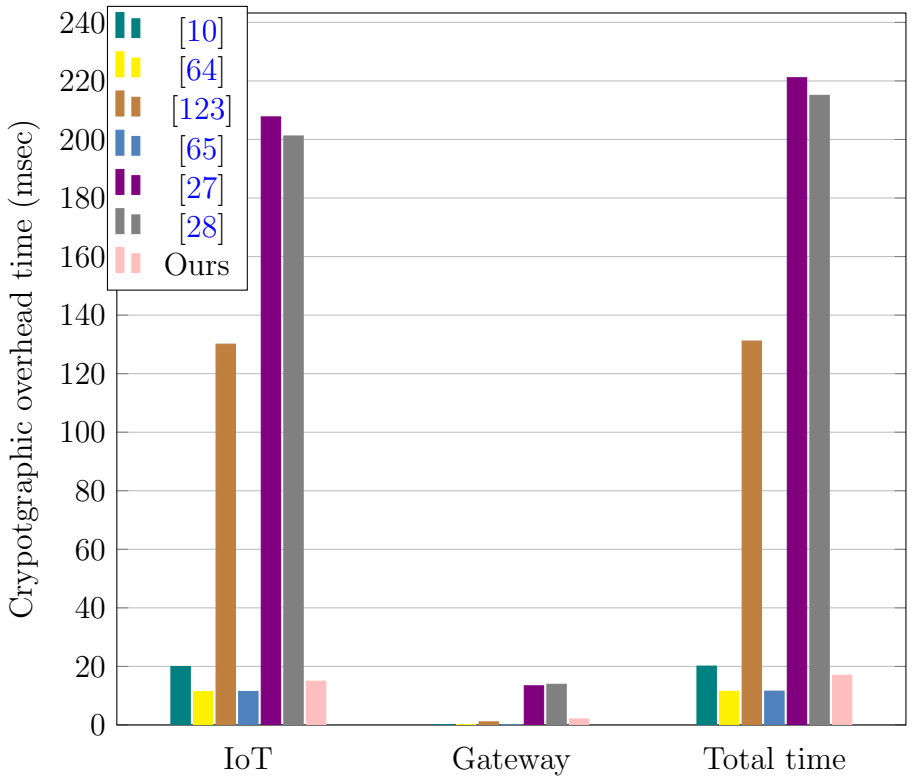

Figure 3.7: The cryptographic overhead time for protocols [10], [64], [123], [65], [27], [28], and SKAFS. The time for the IoT device and the gateway are computed on Raspberry Pi 1 and Raspberry Pi 4, respectively

3.5.3 Computation Cost

Let RNG denote the operation corresponding to invoking the random number generator. In the mutual authentication phase, the IoT device requires 2 RNG, 1 $\mathcal{P}_D(\ )$ call, 2 $\mathcal{P}_F(\ )$ call and 2 hash-based computations. Later, in the synchronization phase, the IoT device requires at most 2 $\mathcal{P}_D(\ )$ calls and 5 hash-based computations. In total, the IoT requires 2 RNG, 3 $\mathcal{P}_D(\ )$ call, 2 $\mathcal{P}_F(\ )$ call and 7 hash-based computations.

In the mutual authentication phase, the IoT gateway performs 2 RNG, 1 encryption operations, 1 decryption operations, 2 fractional hamming computation, and 7 hash-based computations. Alternatively, CA performs 1 decryption operations, and 1 encryption operations. In the synchronization phase, the IoT gateway performs 2 hash-based computation and 1 encryption operation. In total, the IoT gateway requires 2 RNG, 2 encryption

operations, 1 decryption operations, 2 fractional hamming distance computation, and 9 hash-based computation.

3.5.4 Performance Benchmarking and Comparison

To estimate the cryptographic overhead time induced in our protocol, we run the utilized cryptographic primitives in our protocol on a Raspberry Pi 4 Model B/8GB equipped with a 64−bit Quad-core ARM Cortex-A72 processor clocked at 1.5 GHz. Moreover, we consider a Raspberry Pi 1 Model B+ with 512 MB of RAM and a 0.7 GHz ARM11 processor for resource-constrained IoT devices. The average cryptographic time after 1,000,000 runs of the cryptographic primitives on the Raspberry Pi 1 is 0.03376 msec for the Hash, 0.1185 msec for the HMAC, 0.0874 msec for the random number generation, and 0.558 msec for the AES encryption. Table 3.5 shows the average time (msec) on the Raspberry Pi 1 and the Raspberry Pi 4. Moreover, for PUF execution time, we based our calculation on the benchmark in [64] where the execution time of arbiter PUF, SRAM PUF, and DRAM PUF implementation on SASEBO-GII board, consisting of a Xilinx XC5VLX30 FPGA device with a system clock of 1.846 MHz and 16K byte of program memory, is reported to be 3.945, 2.236, and 3.3334, respectively.

Furthermore, we implement SKAFS using socket programming to measure its end-to-end latency and simulate the flow of our protocol messages between the IoT device, IoT gateway, and the CA in a real-time experiment. The Raspberry Pi 1 represents the IoT device while the Raspberry Pi 4 represents the IoT gateway and an Intel laptop 11th Gen Core i7-11800H clocked at 2.3 GHz with 16 GB RAM, acts as the server. On average, the end-to-end authentication protocol incurs a latency overhead of 33.93 ms. These results illustrate the efficiency of the protocol in the IoT-edge-cloud paradigm. Our source code for the Python implementation of the socket programming experiment is available on the GitHub repository at https://github.com/M-Mahdy-Seif/SKAFS.

We compare SKAFS with protocols in [10, 27, 28, 62, 64, 65, 123] in Fig. 6.5, based

Table 3.5: Average cryptographic overhead time (msec)

Description	Notation	Rasp. Pi 1	Rasp. Pi 4
Hash operation [3]	T_h	0.03376	0.001245
MAC operation [4]	T_{MAC}	0.1185	0.00464
RNG	T_{RNG}	0.0874	0.0353
Symmetric Enc/Dec [5]	T_s	0.558	0.03052
Fractional Hamming Distance	FHD	2.26	0.094
Bilinear Pairing operation [6]	T_b	196.29	12.552
Point Addition [7]	T_{PA}	1.199	0.073
Scalar Point Multiplication [5]	T_{PM}	3.20	0.297

The average cryptographic overhead time (msec) after 1,000,000 runs on Raspberry Pi 1 and Raspberry Pi 4.
[1]: We consider SHA-256 over 1024 bytes data
[2]: We consider SHA-256 as the underlying message-digest algorithm
[3]: We consider AES-CBC mode with a key size of 128 bits.
[4]: We use the Python library bplib, which implements a bilinear pairing over a Barreto-Naehrig curve [1].
[5]: Using the fastecdsa Python library.

Table 3.6: Performance comparison based on the computation complexity

Protocol	IoT computation (msec) on Raspberry Pi 1	Gateway computation (msec) on Raspberry Pi 4
[10]	4 APUF + 5 T_{RNG} + 6 T_s + 3 T_{MAC} +2 T_h	4 T_s + 2 T_{MAC} + 2 T_h
[64]	6 T_h + FHD + 2 DPUF + FPUF	T_{RNG} + FHD + 7 T_h
[123]	25 T_{RNG} + 48 FPUF + 163 T_h + 27 T_s	35 T_h + 26 T_s + 48 T_{MAC}
[65]	5 T_h + FHD + 2 DPUF + FPUF + T_{RNG}	T_{RNG} + FHD + 6 T_h
[27]	1 FPUF + 2 T_{PA} + 12 T_h + 2 T_{PM} + 1 T_b	9 T_h + 1 T_b + 2 T_{PM} + 3 T_{PA}
[28]	1 FPUF + 2 T_{PA} + 8 T_h + 1 T_b	8 T_h + 1 T_b + 4 T_{PM} + 2 T_{PA}
Ours	2 T_{RNG} + 3 DPUF + 2 FPUF + 9 T_h	2 T_{RNG} + 2 FHD + 3 T_s + 7 T_h

on the average cryptographic overhead latency, as reported in Table 6.3. The comparison shows that SKAFS outperforms the protocols in [10, 27, 28, 62, 123], and is comparable to those in [64, 65] in terms of execution time. However, the protocols in [10, 64, 65, 123] have higher communication cost than SKAFS as shown in Fig. 3.8.

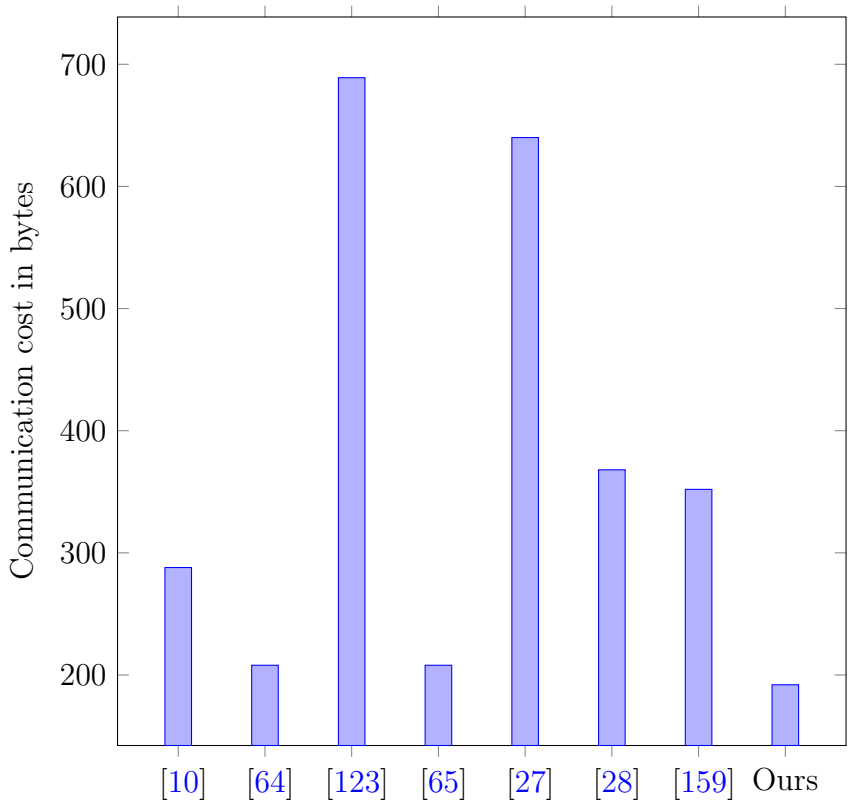

Figure 3.8: Communication cost of [10], [64], [123], [65], [27], [28], [159] and SKAFS

To demonstrate the storage reduction in SKAFS, we assume the synchronization set required in other protocols to thwart DDSA is a set of s pseudo-identities. In Fig. 3.9, we illustrate the scaling of storage requirements of the CA per IoT device with the size of the synchronization set for the protocols [61, 62, 64, 65] as well as SKAFS.

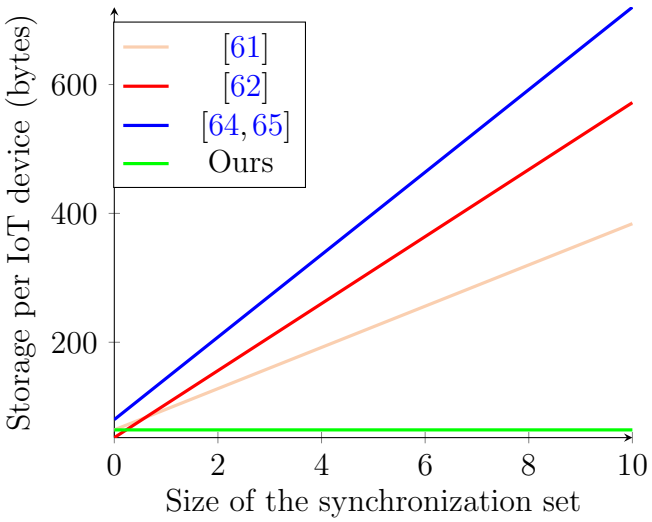

Figure 3.9: The scaling of the storage required at CA per IoT device with the size of the synchronization set

Protocols [65] and [64] require storage of $(80+64\times s)$ bytes per IoT device to keep their identities, keys, and the corresponding challenge-response pairs. On the other hand, in [62], CA needs $(52+52\times s)$ bytes per IoT device, and in [61], a storage of $(64+32\times s)$ is required for every IoT device. Moreover, the protocols in [64] and [65] demand a storage of 48 bytes on the IoT device for keeping the identities, corresponding keys, and challenges. The work in [62] requires $48+s\times 32$ bytes, while that in [61] requires $(182+s\times 174)$ bytes for storing the instantaneous pseudo-identities, challenge, and helper data along with the synchronization identities set. Note that, in the aforementioned protocols, a repeated de-synchronization attack on an IoT device until running out of synchronization identity, requires the IoT device to re-register again with the CA. Nevertheless, SKAFS achieves mutual authentication beside other security properties (i.e., resilience to physical attacks, DDSA, and PUF-modeling attacks) with a constant storage requirement of 64 bytes on the IoT device and storage requirement of 64 bytes/IoT device at CA.

3.6 Summary

In this chapter, we introduced a mutual authentication protocol, named SKAFS, for secure computation offloading services between an IoT device and an IoT gateway in the IoT-edge-cloud paradigm. SKAFS achieve anonymity of the IoT device, forward secrecy, and is resilient to physical and DDSA. SKAFS makes use of PUFs to thwart physical attacks that may lead to compromising the long-term authentication key. We have formally proved that under the assumption of the indistinguishability of secure PUFs, SKAFS is privacy-preserving, and achieves mutual authentication and forward secrecy. Moreover, the synchronization scheme ensures that both the IoT device and the gateway are updated and synchronized by the end of a complete run of the protocol. We have provided performance analysis to show that the protocol is efficient in terms of storage, communication cost, and computation complexity which makes SKAFS compatible with the IoT-edge-cloud paradigm.

Chapter 4

Symmetric Key Inter-Cloud Authentication and Redeemable Micropayment Protocol

4.1 Introduction

While the three-tiered architecture (i.e., IoT-edge-cloud) enables low-end IoT devices to perform with computation capabilities beyond their resources, it gives rise to several security challenges, particularly, for those mobile devices that are expected to be dynamically deployed anywhere such as wearables, and implantable medical devices. More precisely, unlike stationary deployable devices whose cloud admin may assign edge nodes to support the required CO service, mobile ones are likely to require such services when roaming out of their home cloud coverage. Accordingly, previous research has attempted to find efficient solutions to secure roaming services. The work in [146] requires the use of smart cards and a tamper-proof device on the IoT devices which adds extra physical requirements on foreign cloud servers to enable inter-cloud support. Protocols in [38,39] are supported by the blockchain to enable IoT device's payment for CO charges but require

the use of Ethereum accounts and cryptocurrency which may not be applicable for anywhere deployed IoTs that are vulnerable to hardware compromise. The proposal in [60] lacks guaranteed CO charge redemption from foreign service providers, and the authentication protocols in [15, 16, 79, 127] do not provide a mechanism for protecting against hardware compromise.

To address the challenges in the aforementioned protocols and realize secure roaming service for mobile IoTs, we need efficient solutions for the following questions:

- How can we enable mutual authentication between IoT devices and edge nodes that belong to a foreign cloud admin while maintaining the user's privacy to prevent adversarial tracking?
- How roaming charges for computation offloading services are ensured for foreign cloud admins (henceforth referred to by inter-cloud payment)?
- In the event of a device compromise, how can we ensure the security of the stored secrets such that other devices in the network are not affected?

The objective is to realize a secure delegation-based authentication protocol by finding a solution to the raised questions that enables roaming IoTs to obtain CO services from foreign cloud admins. We propose SKICAP, a privacy-preserving delegation-based authentication and hardware-compromise resilient payment protocol. The protocol makes use of PUF to compute the authentication keys instead of digitally storing them on the IoT device and being vulnerable to physical attack. To address the three challenges listed in the motivation part, the home cloud admin, where the IoT device subscribes, shares only two secret keys with other foreign cloud admins to derive a tree of secrets for decoding and authenticating the foreign anonymous IoT devices. Additionally, upon requesting CO service from the foreign cloud admin, the IoT device provides a payment token and membership proof to the hosting cloud admin. The hosting cloud admin verifies the membership proof of the payment token to ensure the subscription for CO service with

the home cloud. A comparison of SKICAP with other related works adopting the IoT-Edge-Cloud paradigm is summarized in Table 4.1.

Table 4.1: Comparison with other related IoT-Edge-Cloud protocols.

Security Properties	[146]	[77]	[79]	[15]	[16]	[104]	[107]	[29]	[13]	SKICAP
Mutual Authentication	✓	✓	✓	✓	✓	✓	×	✓	✓	✓
Confidentiality	✓	✓	×	✓	×	✓	✓	✓	✓	✓
Unlinkability	✓	✓	✓	✓	✓	×	×	✓	✓	✓
Traceability	✓	✓	✓	✓	✓	✓	×	×	×	✓
Inter-Cloud Payment Support	×	×	×	×	×	✓	×	×	×	✓
Hardware Compromise Resilience	✓	×	×	×	×	×	×	×	×	✓

4.2 System Model, Entities, and Design Goals

In this section, we present our system model along with the entities in our system. Then, we present the threat model and design goals of our protocol. Throughout the chapter, index $h \in \{0, 1, 2, , ..., N-1\}$ is in decimal representation where N is the number of registered IoT devices. For the illustration of figures, we use the binary representation for the index h.

4.2.1 System Model

As shown in Fig. 4.1, our system model adopts a three-tier hierarchy system which is composed of a cloud server, IoT gateways, (e.g., routers, switches, and proxy servers), and IoT devices (e.g., wearable devices in healthcare systems or virtual reality helmets). IoT gateways provide CO services to IoT devices over the wireless link. Moreover, IoT gateways are securely connected to their cloud server. IoT devices are equipped with PUF and any external access to the PUF circuit changes its challenge-response behavior. The PUF and the microcontroller are considered to be a system-on-chip and the connection between them is considered to be secure in the sense that a probabilistic polynomial time adversary cannot intercept the communication between the PUF and the microcontroller

[10, 59].

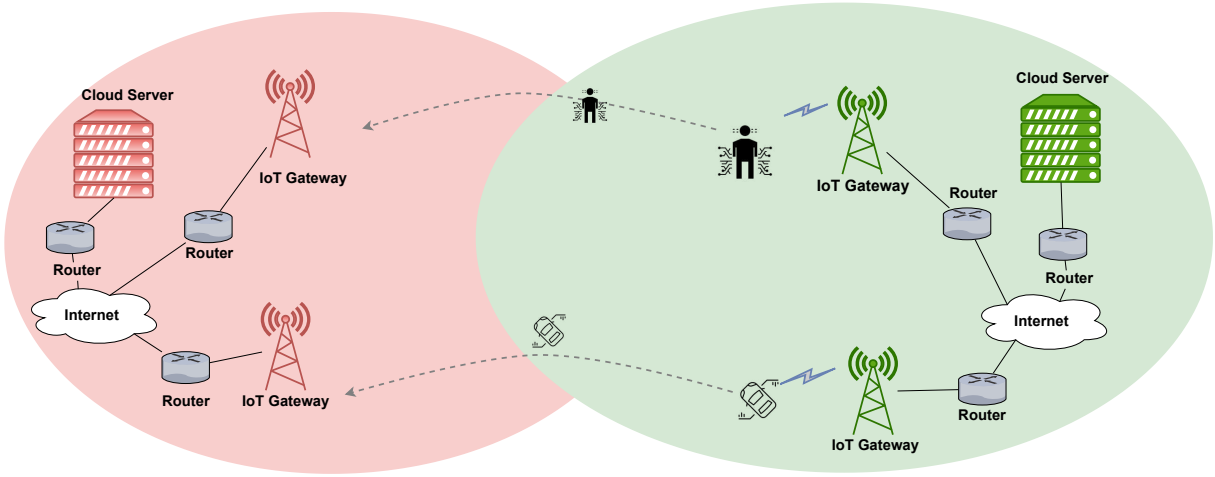

Figure 4.1: An illustration of the considered IoT-edge-cloud paradigm. The dashed lines represent the transition of an IoT user from the subscribed home cloud coverage area, depicted in green, to the hosting cloud coverage area, depicted in red

The entities in our model are:

- Cloud Admin: The cloud admin is deployed by the cloud service provider and is responsible for the registration of IoT devices and managing the subscriptions of IoT-Edge CO services. Additionally, the home cloud admin shares with foreign cloud admins the secrets S and K_0 to easily authenticate IoT devices. Moreover, it shares a Merkle tree root to verify the validity of the subscription tokens of IoT devices.

- IoT Gateway: Computation-capable IoT gateways are deployed by the cloud service provider. An IoT gateway can be a router, a switch, or a proxy server. In our model, we assume IoT gateways are securely connected to the cloud server.

- IoT Devices: The IoT devices are resource-constrained devices with internet accessibility. They are enrolled in many applications like healthcare and virtual reality. Occasionally, these devices need to outsource some of their heavy computation and storage to computation-capable IoT gateways.

4.2.2 Threat Model

We consider the traditional Dolev-Yao model [44] where the communicating entities (i.e., IoT device and IoT gateway) use a public channel which is considered insecure. An adversary $\mathcal{A}$ over the wireless channel between the IoT device and the IoT gateway can eavesdrop on the communication channel. In addition, $\mathcal{A}$ can drop, insert, and modify messages to the protocol. We define the additional capabilities of an adversary against our protocol in the form of oracle queries in Section 4.4.1. In what follows, we list our threat assumptions concerning IoT devices, IoT gateways, and cloud admins

- IoT devices: We assume malicious IoT devices that may attempt to obtain CO services without a valid subscription, and/or try to impersonate other devices.
- Cloud admins and IoT gateways: We assume that both the cloud admins and IoT gateways behave honestly and that they are securely connected.

4.2.3 Design Goals

The design goals of our proposed protocol can be summarized as follows:

- Mutual Authentication: Verifying the legitimacy of both the IoT device and the IoT gateway of the hosting cloud.
- Message Confidentiality and Integrity: An external adversary cannot compromise the confidentiality or the integrity of messages between the IoT device and the IoT gateway.
- Hardware Compromise Resilience: The compromise of an IoT device does not lead to leakage of the key materials needed for deriving the authentication of other IoT devices.
- Unlinkability: An external adversary cannot correlate authentication requests of

any IoT device. The IoT device's authentication request should be indistinguishable from a pseudo-random sequence.

- Guaranteed Token Redemption: The IoT gateway of the hosting cloud has a verifiable valid payment token for the CO service offered to the requesting IoT device. Later on, the foreign hosting cloud redeems the CO charges from the home cloud admin who verifies that this token covers the IoT device subscription.

- Traceability: In case of misbehavior from an IoT device, the home cloud admin can trace the temporary pseudo-identity to its real identity.

- Efficiency: SKICAP should not rely on complex public key cryptographic operations as these operations require computation power which may not be affordable by resource-constrained IoT devices. Instead, SKICAP utilizes lightweight primitives, such as hash-based computations and symmetric-key encryption, to be compatible with limited-resource IoT devices.

4.3 Protocol Design and Implementation

We consider two cloud admins: a home cloud admin CA_h where IoT devices register and subscribe, and a hosting cloud admin CA_v. When an IoT device IoT_h registered with CA_h moves to another district out of the coverage service of CA_h but with some deployed IoT gateways for CA_v, the IoT gateway of the hosting cloud authenticates the IoT device IoT_h. To validate the IoT subscription with the home cloud admin, IoT_h provides a verifiable payment token and a subscription proof with the home cloud admin to the hosting cloud admin CA_v. Before serving the IoT device, the hosting cloud admin verifies that the payment token covers the required CO units by the IoT device and validates its subscription proof. Then, in the end, during the redemption interval, CA_v collects the tokens from the serving IoT gateways and redeems the CO charges from CA_h.

4.3.1 Protocol Overview

In this section, we present the design of our proposed protocol and explain the various details regarding its implementation. SKICAP makes use of hash chains [87], tree of secrets [105], Merkle trees [103] and Physical Uncolneable Functions [99]. A hash chain is constructed by repeatedly hashing a random value $\mathcal{S}$, the seed of the hash chain. Let $H^{i+1}(\mathcal{S}) = H(H^i(\mathcal{S}))$, thus, a hash chain of length $n+1$ is ($\mathcal{S}$, $H(\mathcal{S})$, $H^2(\mathcal{S})$, ..., $H^{n-1}(\mathcal{S})$, $H^n(\mathcal{S})$) where $\mathcal{S}$ is the hash chain seed and $\mathcal{T} = H^n(\mathcal{S})$ is the tip of the hash chain. Merkle tree is a way of accumulating data, where any modification of the data results in a new Merkle root. SKICAP runs as follows. Initially, IoT devices register with the cloud admin CA_h using their real identities. CA_h builds a tree of secrets to obfuscate the IoT device's identities. When an IoT device subscribes for L CO units, CA_h generates a hash chain for the IoT device. Then, CA_h accumulates tips of all the hash chains in a Merkle tree. CA_h shares with other foreign cloud admins the secrets S and K_0 for deriving the tree of the secrets and the Merkle tree root. Then, CA_h sends to each IoT device the seed of its hash chain, and the Merkle proof of its tip in the CA_h Merkle tree. Moreover, for each IoT device, CA_h stores a set of keys that represents a path from the root to a leaf in the tree of secrets. This set of keys is required for authenticating the IoT device. Each key is digitally stored in the IoT device as PUF calls for two different input challenges. In the mutual authentication phase, the IoT device sends a temporary pseudo-identity to the IoT gateway as a response to the IoT gateway challenge and mutual authentication between the IoT device and the IoT gateway is carried. After the authentication, the IoT device and the IoT gateway derive a session key to encrypt the messages between them. Then, the IoT device sends the number of the needed computation units n and a pre-image value in its hash chain as a payment token. Moreover, for verifying the payment token, the IoT device sends the tip of its hash chain and the proof of the membership of the tip in the CA_h Merkle tree. In the redemption phase, the IoT gateway of the hosting CA sends the collected tokens to the CA_h to redeem the expenses of the CO units offered to the IoT_h.

Registration and Set-up Phase

CA_h generates the random keys S, K_0 and K_1 where S and K_0 are two secret keys used to generate the tree of secrets and K_1 is a key known only by CA_h. Subsequently, CA_h builds up a tree of secrets for the registered IoT devices as shown in Fig. 4.2. The root of the tree is S which has two child nodes, $S_0 = H_{K_0}(0||S)$ and $S_1 = H_{K_0}(1||S)$, respectively, $H_{K_0}(\ )$ is a keyed hash function under the key K_0 where the key K_0 is known only by CA_h and other foreign cloud admins. Similarly, node S_0 has two child nodes, $S_{00} = H_{K_0}(0||S_0)$ and $S_{01} = H_{K_0}(1||S_0)$, and so on. For each IoT_h, CA_h assigns a set of keys that represents a path from the root to a leaf in the tree of secrets (e.g., for IoT_{010} of index $h = 010$, CA_h assigns the set of keys $S_0,\ S_{01},\ S_{010}$).

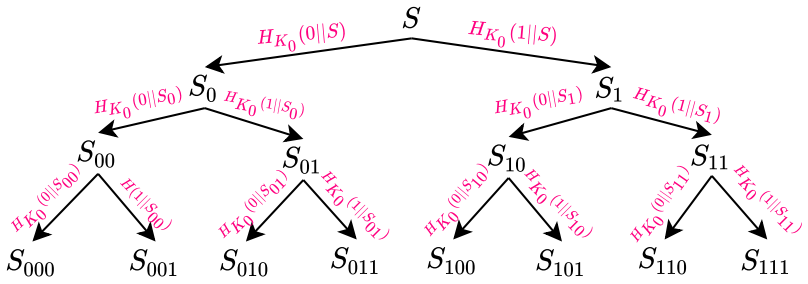

Figure 4.2: An example of a tree of secrets for eight IoT devices pseudo-identities

These secret keys are used later to generate temporary pseudo-identities of IoT_h for authentication with the IoT gateway of CA_v. Additionally, for each IoT device, the cloud admin CA_h creates a hash chain with seed $\mathcal{S}_i = H(K_1||S_i)$ and tip $\mathcal{T}_i = H^L(\mathcal{S}_i)$ where L is the length of the hash chain. Then, CA_h accumulates the tips of all the hash chains in a single Merkle tree [103]. Fig. 4.3 shows an example for accumulating the tips of 8 IoT devices in a single Merkle tree.

CA_h sends to each IoT device $\mathcal{S}_i,\ \mathcal{T}_i,\ \pi_i$ where $\mathcal{S}_i$ is the seed of the IoT device hash chain and $\mathcal{T}_i$ is the tip of the hash chain and π_i is the proof of membership of $\mathcal{T}_i$ in the CA_h Merkle tree (e.g., for IoT_{010}, the proof is $\pi_{010} = (H(\mathcal{T}_{011}),\ H_{00},\ H_1)$) as shown in 4.4. Instead of storing the set of keys in the IoT device memory, the IoT device stores only two

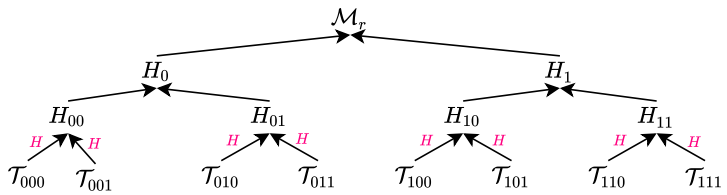

Figure 4.3: Merkle tree of the tips of a hash chain of eight IoT devices

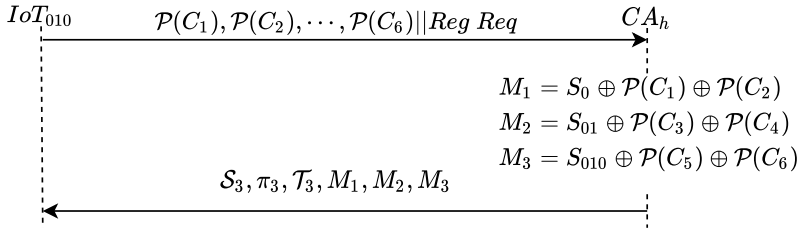

Figure 4.4: Registration of the IoT device, IoT_{010}

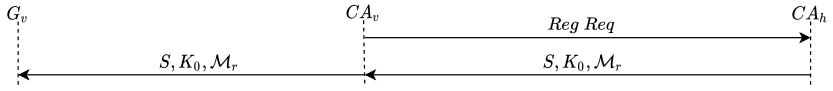

Figure 4.5: Registration of the IoT gateway G_v

PUF challenges for each key such that $\mathcal{K}_i = \mathcal{P}(C_{2i-1}) \oplus \mathcal{P}(C_{2i}) \oplus M_i$ where M_i is the result of bitwise XOR operation between $\mathcal{K}_i$ and the PUF calls for the two input challenges (e.g., for IoT_{010}, CA_h assigns the set of keys $\mathcal{K}_1 = S_0$, $\mathcal{K}_2 = S_{01}$, $\mathcal{K}_3 = S_{010}$. The keys are stored on the IoT device as $\mathcal{P}(C_1) \oplus \mathcal{P}(C_2) \oplus M_1$ for $\mathcal{K}_1$, $\mathcal{P}(C_3) \oplus \mathcal{P}(C_4) \oplus M_2$ for $\mathcal{K}_2$ and $\mathcal{P}(C_5) \oplus \mathcal{P}(C_6) \oplus M_3$ for $\mathcal{K}_3$). An example of registering an IoT device, IoT_{010}, is depicted in Fig. 4.4 where the IoT device sends a registration request along with the PUF responses $\mathcal{P}(C_1), \mathcal{P}(C_2), ..., \mathcal{P}(C_6)$ to the cloud admin. Then, the cloud admin responds with M_1, M_2, M_3 along with the tip $\mathcal{T}_3$, Merkle proof π_3 and pseudo-identity $\mathcal{S}_3$. For reducing the storage of the challenges in the IoT device storage, challenges are chained such that $C_{i+1} = H(C_i)$ where i starts from 1 and only C_1 is stored on the IoT device.

On the other side, CA_h sends to foreign cloud admins the secrets S, K_0 and the Merkle root $\mathcal{M}_r$ as illustrated in Fig. 4.5. CA_v uses the secret key S and K_0 to decode the temporary pseudo-identities of IoT devices in the mutual authentication phase, while $\mathcal{M}_r$ is used for verifying the Merkle proof of the tip $\mathcal{T}_i$.

Mutual Authentication and Key Agreement Phase

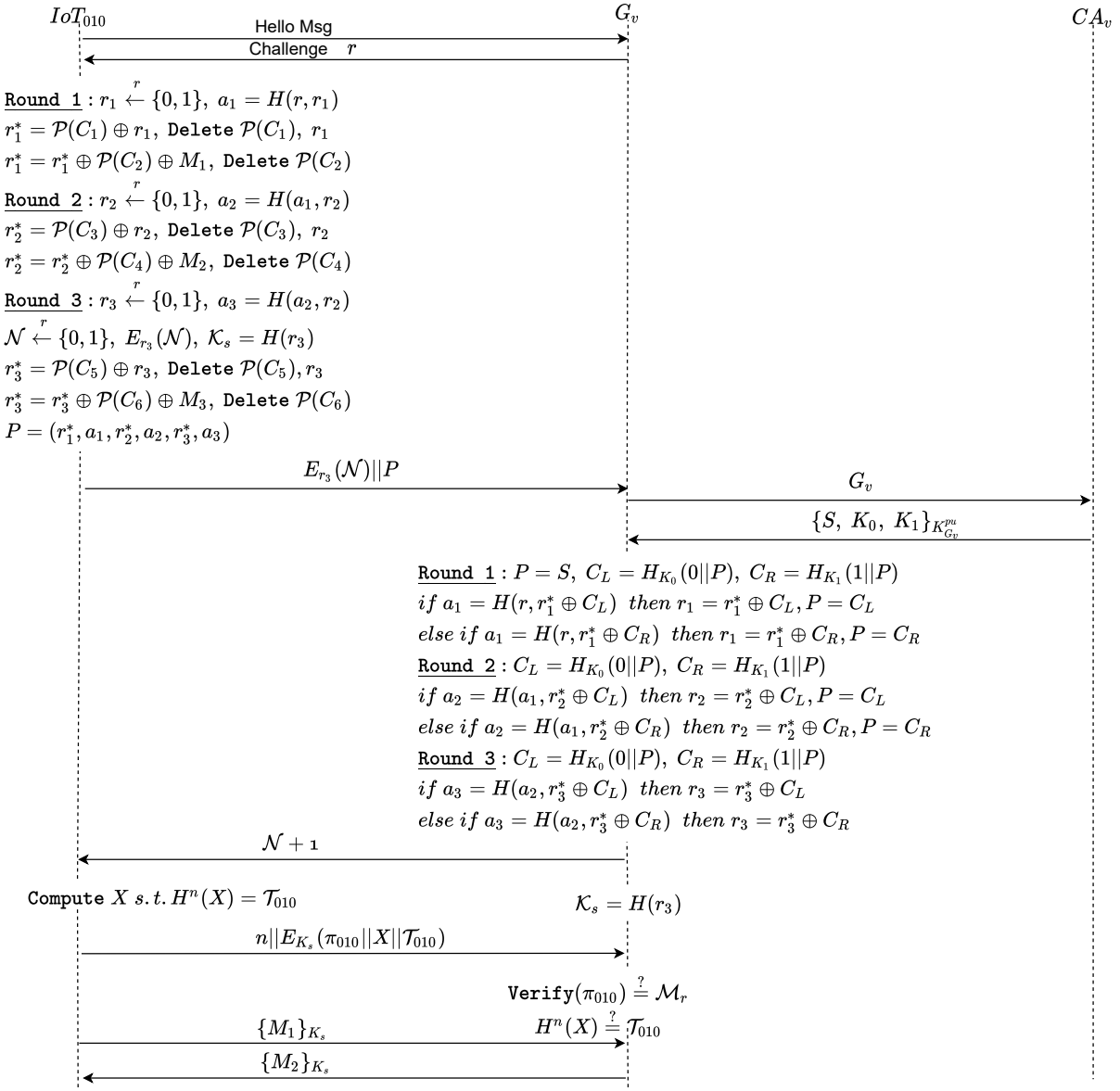

Figure 4.6: Mutual authentication between IoT_{010} and the IoT-Gateway G_v of CA_v

The mutual authentication between IoT_h, depicted as IoT device IoT_{010} in Fig. 4.6, and the IoT gateway G_v of CA_v starts by sending a Hello message to the IoT gateway. Then, the IoT gateway challenges the IoT device with a random r. Next, the IoT device generates a random number r_1 and computes the authentication message $a_1 = H(r, r_1)$. After that, to obfuscate r_1 with the first key in its set of keys that represents a path in

the tree of secret, the random r_1 is bitwise XORed with the PUF response for the stored challenge C_1. The result of the bitwise XOR operations is bitwise XORed with the PUF response for C_2 and M_1 such that $\mathcal{P}(C_1) \oplus \mathcal{P}(C_2) \oplus M_1 = \mathcal{K}_1$, where $\mathcal{K}_1$ is the first key along the path in the tree of secret for the IoT device, e.g., S_0 for IoT_{010}. Similarly, the IoT device obfuscates its randoms r_2, r_3, ..., r_z using the keys $\mathcal{K}_2$, $\mathcal{K}_3$, ..., $\mathcal{K}_z$ where $z = Log(N)$ and represents the depth of the tree of secrets ($z = 3$ in our illustrated example). In the last key round, the IoT device encrypts the nonce $\mathcal{N}$ using the random r_z and precomputes the session key $\mathcal{K}_s = H(r_z)$. Then, the IoT device responds to the IoT gateway challenge with the temporary pseudo-identity $P = \{(r_1^*, a_1), (r_2^*, a_2), ..., (r_z^*, a_z)\}$ and the encrypted nonce $E_{r_z}(\mathcal{N})$. When the IoT gateway receives P, the IoT gateway decodes the temporary pseudo-identity of the IoT device starting from the top, root S, and proceeds downward by doing the decryption of r_1^* using the keys in the first level to get r_1 and checking $a_1 \stackrel{?}{=} H(r, r_1)$. The IoT gateway follows the matching path until it reaches the corresponding pseudo-identity of the IoT device, (S_{010} in our illustrated example). Subsequently, the IoT gateway increments the IoT device nonce, $\mathcal{N}$, and sends it back to the IoT device, IoT_{010}. Then, the IoT device and the IoT gateway agree on the session key $\mathcal{K}_s = H(r_z)$ and the IoT device sends the payment X for n required computation offloading services such that $H^n(X) = \mathcal{T}$. After that, the IoT device sends n and $E_{\mathcal{K}_s}(\pi, X, \mathcal{T})$, the encryption of the Merkle proof, the payment token and the tip of the IoT hash chain. The IoT gateway verifies the payment token X by checking $H^n(x) = \mathcal{T}$. Also, the IoT gateway verifies the Merkle proof π of the payment token X using the algorithm in [52]. Then, the IoT device and IoT gateway exchange the message encrypted using an authenticated encryption scheme with the session key $\mathcal{K}_s$. Note that, for the sake of clarification and clear separation between mutual authentication and message exchange phases, we send the encrypted messages $\{M\}_{K_s}$ in a separate step. However, if required for decreasing the number of the protocol rounds, the encrypted messages $\{M\}_{K_s}$ can be sent along with the number of the computation units and the encryption of the Merkle

proof, payment token, and the tip in one step.

CO Charges Redemption Phase

At the end of each redemption interval, CA_v sends the set of the pseudo-identities of the IoT device $\mathbf{S}$ and the set of the payment tokens $\mathbf{X}'$ to CA_h as shown in Fig. 4.7. To verify the token and pay back the CA_v, the home cloud initializes a counter $c = 0$; then it repeatedly hashes the seed until it is equal to the payment token X'_i. If the counter reaches L without equality, CA_h considers it as an invalid payment token. Otherwise, CA_h pays back the charges of $(n' = L - c)$ CO units to CA_v.

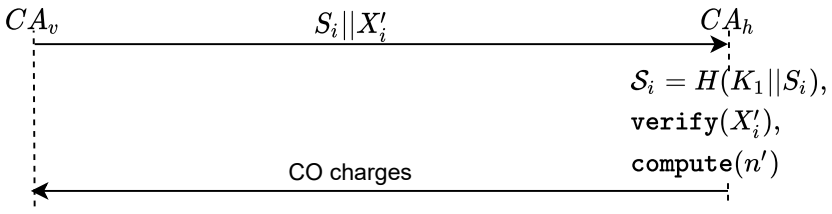

Figure 4.7: Charges redemption for claimed token X' in return for n' CO units from the hosting cloud

4.3.2 Revocation of gateways

In what follows, we show how SKICAP can solve the revocation if some IoT gateways get compromised. For this, we require the home cloud admin CA_h to share with the registered IoT devices the secret K_h. An embedded PUF circuit is used for storing the key K_h on the IoT device and thwarting physical attacks on the IoT device [55]. The home cloud CA_h updates its tree of secrets such that the updated tree secrets become $S^{'} = E_{K_h}(S||U_v)$, $S^{'}_0 = E_{K_h}(S_0||U_v)$, and so on, where $S^{'}$ is the new updated secret value and S is the initial secret value. Then, CA_h sends the updated value U_v and its encryption $U^{'}_v = E_{K_h}(U_v)$ to other hosting cloud admins CA_v. The updating of the tree is done on a timely basis. When an IoT device requires a computation offloading service from a non-compromised IoT gateway, the IoT gateway broadcasts the encrypted $U^{'}_v$ obtained

from the hosting cloud admin to this IoT device. Then, the IoT device generates its new temporary pseudo-identity using updated keys along its path $\mathcal{K}'_i = E_{K_h}(\mathcal{K}_i||U_v)$ where $\mathcal{K}'_i$ is the new IoT key for round i. In such a scenario, a compromised IoT gateway does not obtain this U_v, hence it will not be able to achieve mutual authentication or decode the IoT device pseudo-identity. Note that our solution follows the model introduced by short-lived certificate [34], where the protocol tolerates an allowable interval of gateway compromise, i.e., from the time a gateway is compromised until the scheduled update time.

4.4 Security Analysis

We start this section by sketching some possible actions of the adversary who tries to attack the protocol. Then, we define the completeness and soundness of our protocol. After that, under the assumption of a secure hash function and an indistinguishably secure PUF, we proceed with the formal security proof of our protocol.

4.4.1 Adversary Model

Under the Dolev-Yao threat model discussed in Section 6.2.2, we consider an adversary $\mathcal{A}$ who has control over the wireless channel between the IoT device and the IoT gateway. In addition, $\mathcal{A}$ can make the following oracle queries.

- **Launch**(1^λ)**:** Establishes a new protocol instance with identifier π.
- **Send Gateway( ):** Models the adversary interactions with the IoT gateway where $\mathcal{A}$ acts as a legitimate IoT device. $\mathcal{A}$ gets the challenge r from the IoT gateway and possibly responds with the temporary pseudo-identity P. Nevertheless, $\mathcal{A}$ does not receive the result of the identification.
- **Send IoT( ):** Models the adversary interactions with an IoT device where $\mathcal{A}$ acts

as a legitimate IoT gateway. In this context, $\mathcal{A}$ sends a challenge r to the IoT device, and the IoT device responds with its temporary pseudo-identity P.

- **Result( ):** Returns whether an input transcript $\mathcal{T}_r$ is a valid communication transcript of the protocol or not. It gives the authentication result for the challenge r from the IoT gateway and temporary pseudo-identity P from the IoT device.

- **Reveal( ):** Models the adversary's capability to open an IoT device and obtain the information stored in its memory. Similar to [22, 61, 76], we assume the model which is proposed in [55] where the author has shown that a physical attack on the memory is always invasive and ultimately leads to the destruction of its on-chip neighboring PUF. Accordingly, our **Reveal** oracle can be invoked once on the IoT device and the adversary cannot use the output of the PUF afterward.

4.4.2 Security Analysis

In what follows, we define the completeness, soundness, and privacy of our protocol. Then, we illustrate how SKICAP addresses the three questions raised in the introduction. Under the assumption of a secure PUF and a secure hash function, we show that SKICAP is complete and sound, and maintains the privacy of the IoTs. We also prove that compromising an IoT device leaks no additional information on the key materials.

Definition 4.1 (Completeness) *SKICAP is said to be complete if a legitimate IoT device has a negligible probability to fail in the authentication process, formally, for all legitimate IoTs,* $\mathsf{Pr}[\boldsymbol{Result}(r_0, \boldsymbol{Send\ IoT}(r_0)) = false \mid r_0 = \boldsymbol{Send\ Gateway(\ )}] \leq \epsilon$.

Definition 4.2 (Soundness) *SKICAP is said to be sound if a probabilistic polynomial time adversary has a negligible probability to produce a valid communication transcript* $\mathcal{T}_r$*, formally,* $\mathsf{Pr}[\boldsymbol{Result}(\mathcal{T}_r) = true] \leq \epsilon$.

Theorem 4.1 *Compromising an IoT device and getting access to the IoT device storage does not leak any additional information on the key materials.*

Proof: Herein, we prove that an adversary $\mathcal{A}$ with access to the **Reveal** oracle gets information on the key material as he would get from the **Send Gateway** oracle. An adversary $\mathcal{A}$ can compromise an IoT device during the first $z-1$ rounds or the last round. $\mathcal{A}$ can compromise the IoT device before the first PUF call to get the generated random r_i or after the first PUF call to get the random r_i and $r_i \oplus \mathcal{P}(C_{2i-1})$ or after the second PUF call to get $r_i \oplus \mathcal{P}(C_{2i-1}) \oplus \mathcal{P}(C_{2i})$. Compromising the IoT device in the last round is similar to the previous round in addition to leaking $E_{r_z}(\mathcal{N})$ and $H(r_z)$. Therefore, compromising an IoT device leads the adversary to have access to one of the following sets of information:

- $M_i,\ a_i,\ r_i$ and a_{i+1}.
- $M_i,\ a_i,\ r_i,\ a_{i+1}$ and $r_i \oplus \mathcal{P}(C_{2i-1})$.
- $M_i,\ a_i,\ r_i \oplus \mathcal{P}(C_{2i-1}),\ a_{i+1}$ and $r_i \oplus \mathcal{K}_i$.
- $M_z,\ a_{z-1},\ r_z,\ a_z,\ r_z \oplus \mathcal{P}(C_{2z-1}),\ H(r_z)$ and $E_{r_z}(\mathcal{N})$.
- $M_z,\ a_{z-1},\ r_z \oplus \mathcal{P}(C_{2z-1}),\ a_z,\ r_z \oplus \mathcal{K}_z,\ H(r_z)$ and $E_{r_z}(\mathcal{N})$.

Since $\mathcal{K}_i = \mathcal{P}(C_{2i-1}) \oplus \mathcal{P}(C_{2i}) \oplus M_i$, getting $M_i,\ a_i,\ r_i$ and a_{i+1} leaks no information about the key materials of the IoT device. Getting $M_i,\ a_i,\ r_i\ ,\ a_{i+1}$ and $r_i \oplus \mathcal{P}(C_{2i-1})$ is equivalent to getting $M_i,\ a_i,\ r_i$ and $\mathcal{P}(C_{2i-1})$. Let l_r denote the length of the PUF response and $\mathcal{P}(C_{2i-1})$ is a random value of entropy l_r. Since $\mathcal{K}_i$ of the IoT device equals $\mathcal{P}(C_{2i-1}) \oplus \mathcal{P}(C_{2i}) \oplus M_i$, therefore, the key $\mathcal{K}_i$ is still a random value of entropy l_r. Similarly, getting $a_i,\ r_i \oplus \mathcal{P}(C_{2i-1}),\ a_{i+1}$ and $r_i \oplus \mathcal{K}_i$ is equivalent to getting $a_i,\ \mathcal{P}(C_{2i}),\ a_{i+1}$ and $r_i \oplus \mathcal{K}_i$. Therefore, the key $\mathcal{K}_i$ remains a random value of entropy l_r. Corrupting the IoT device in the last round leads to compromising either $M_z,\ a_{z-1},\ r_z,\ a_z,\ r_z \oplus \mathcal{P}(C_{2z-1}),\ H(r_z),\ \mathcal{N}$ and $E_{r_z}(\mathcal{N})$ or $M_z,\ a_{z-1},\ r_z \oplus \mathcal{P}(C_{2z-1}),\ a_z,\ r_z \oplus K_z,\ H(r_z),\ \mathcal{N}$ and $E_{r_z}(\mathcal{N})$. The

additional information $H(r_z)$, $\mathcal{N}$ and $E_{r_z}(\mathcal{N})$ in the last round is of no interest in the first case. Also, in the second case of the last round, under the assumption of a preimage-resistant hash function and a secure encryption system, the additional information is of no value to the attacker and the key $\mathcal{K}_z$ remains a random value of entropy l_r. ■

Theorem 4.2 *SKICAP achieves mutual authentication.*

Proof: We show the mutual authentication property of SKICAP by proving the completeness of the protocol where a legitimate IoT device has a negligible probability of failing in the mutual authentication process. Moreover, we prove the security of SKICAP against impersonation attacks by proving the soundness of the protocol where a non-legitimate user has a negligible property in producing a valid authentication transcript.

For ensuring the completeness property of SKICAP, the IoT gateway traverses the tree of secrets from top to down, level by level, and in each level the IoT gateway checks if the equality $a_i \stackrel{?}{=} H(a_{i-1}, r_i)$ holds for any of the two branches. Without loss of generality, assume that in the i-th level, the IoT gateway starts by checking the left branch and the legitimate user path is on the right branch. Hence, in this case, if $H(a_{i-1}, r_i|_L)$=$H(a_{i-1}, r_i|_R)$, an error occurs because the IoT gateway will proceed with the left branch where $r_i|_L$ and $r_i|_R$ are the random variables of the IoT device with authentication path concurrent with the left branch and right branch, respectively, in the i-th level.

Consider the event $\mathbb{A}$ where the triplet $(a_{i-1},\ H(a_{i-1}, r_i),\ r_i \oplus \mathcal{K}_i)$ of the right IoT device collides with the left IoT device. Event $\mathbb{A}$ occurs when the equality $H(a_{i-1}, r_i|_R)$= $H(a_{i-1}, r_i|_R \oplus \mathcal{K}_i|_R \oplus \mathcal{K}_i|_L)$ holds where $\mathcal{K}_i|_L$ and $\mathcal{K}_i|_R$ are the authentication keys for the left and right IoT devices at the i-th level. The probability of this event $\mathbb{A}$ is $\frac{1}{2^{l_h}}$ where l_h is the length of the output of the hash function. Therefore, an error in authentication in the i-th level could appear with probability of $P = \frac{1}{2^{l_h}}$. Let z be the depth of the tree of secrets where $z = Log(N)$ and N is the number of subscribed IoT devices. Therefore,

the total probability of an unsuccessful authentication equals $\sum_{i=1}^{z} P(1-P)^{i-1}$ where $P(1-P)^{i-1} \approx P$ is the probability of an unsuccessful authentication in the i-th level. Accordingly, the total probability of an unsuccessful authentication is approximately $zP = \frac{z}{2^{l_h}}$.

For the soundness of the protocol, we prove the soundness of SKICAP for one level of the tree of secrets and as the depth of the tree increases, the difficulty of the authentication increases (i.e., the success probability of an illegitimate IoT device in producing a valid authentication will decrease). Let M is the maximum number of the operation that can be made by an adversary within the authentication time. Let the adversary $\mathcal{A}$ intercept the challenge r_0, the probability of the adversary $\mathcal{A}$ to produce a valid couple $(a_1, r_1 \oplus \mathcal{K}_1)$ is $\frac{2M}{2^{l_h}}$. Therefore, the probability for producing a valid communication transcript $\mathcal{T}_r$ by an adversary $\mathcal{A}$ is $(\frac{2M}{2^{l_h}})^z$ where z is the depth of the tree. ■

Theorem 4.3 *SKICAP achieves anonymity.*

Proof: The anonymity of SKICAP is maintained if a probabilistic polynomial time adversary who tries to break the anonymity of the protocol by linking the temporary pseudo-identities of an IoT device has a negligible success probability. This can be achieved when the temporary pseudo-identity is indistinguishable from a random sequence. In the following, we define an anonymity game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ who wants to break the anonymity of the transacting IoT. The anonymity game runs as follows:

- In the setup phase, $\mathcal{C}$ initializes the system with two IoT devices IoT_1 and IoT_2 and the IoT gateway G_v.
- In the learning phase, $\mathcal{A}$ does the following:
  1. $\mathcal{A}$ perfoms **Send IoT** queries on IoT_1 and IoT_2.
  2. $\mathcal{A}$ performs **Send Gateway** query on the IoT gateway G_v.
  3. $\mathcal{A}$ performs **Result** query.

- After the learning phase, $\mathcal{A}$ notifies the challenger $\mathcal{C}$.
- In the challenge phase, the challenger $\mathcal{C}$ samples the bit $b \xleftarrow{r} \{0,1\}$ and selects the IoT device IoT_b.
- $\mathcal{A}$ does the following:
    1. $\mathcal{A}$ performs **Send IoT** query on IoT_b.
    2. $\mathcal{A}$ performs **Send Gateway** query on the IoT gateway G_v.
    3. $\mathcal{A}$ performs **Result** query.
    4. $\mathcal{A}$ performs **Reveal** query on IoT_b.
- $\mathcal{A}$ outputs his guess b' for the sampled bit b and wins the game if $b = b'$.

The advantage of $\mathcal{A}$ is defined as $adv_{\mathcal{A}}^{anony} = |\mathsf{Pr}[b' = b] - 1/2|$. The transacting IoTs are anonymous if for any probabilistic polynomial adversary $\mathcal{A}$ cannot distinguish between the temporary pseudo-identities of two IoT devices (i.e., $adv_{\mathcal{A}}^{anony} \leq \epsilon$). Let us assume an adversary $\mathcal{A}$ who can distinguish between temporary pseudo-identities from two different IoT devices. This means that this adversary can distinguish between an IoT device's temporary pseudo-identity and a pseudo-random sequence. Then, such an adversary $\mathcal{A}$ can be used as a subroutine in adversary $\mathcal{D}$ to break the indistinguishability property of PUFs. This can be achieved as follows. The adversary $\mathcal{D}$ intercepts the temporary pseudo-identity and gets $a_{i-1},\ a_i,\ r_i^*$ from the IoT device IoT_b where $a_i = H(a_{i-1,\, r_i})$, $r_i^* = r_i \oplus K_i$ and $K_i = \mathcal{P}(C_{2i-1}) \oplus \mathcal{P}(C_{2i}) \oplus (M_i)$. Then, the adversary $\mathcal{D}$ uses $\mathcal{A}$ to invoke the **Reveal** oracle on the IoT_b to get M_i, C_{2i} and C_{2i-1}. Then, in the PUF indistinguishability game, in the learning phase, adversary $\mathcal{D}$ gets the PUF response $\mathcal{P}(C_{2i-1})$ for the challenge C_{2i-1}. Later, in the challenge phase, the adversary D submits the challenge C_{2i} and gets the response R_b. D checks $a_i \stackrel{?}{=} H(a_{i-1}, r_i^* \oplus \mathcal{P}(C_{2i-1}) \oplus R_b \oplus M_i)$. $\mathcal{D}$ wins the PUF indistinguishability game with a non-negligible probability by outputting $b = 1$ if the equality holds and $b = 0$, otherwise. Since we assumed a secure indistinguishable PUF,

such an adversary D does not exist and this implies the non-existence of the adversary A and SKICAP maintains the anonymity property. ■

4.4.3 SKICAP Freshness, Guaranteed Redemption, and Traceability

Replay Attack Resilience

An attacker may replay previous packets to launch a denial-of-service attack over the IoT gateway. However, SKICAP counters this attack by incurring the challenge r from the IoT gateway, and randoms $r_1, r_2, ..., r_z$ from the IoT device. In the mutual authentication phase between the IoT device and IoT gateway, the authentication messages are chained where $a_i = H(a_{i-1}, r_i)$ and $a_1 = H(r, r_1)$. Therefore, under the assumption of a secure and collision-resistant hash function, any replay attack using a pre-transmitted authentication message is detected.

Guaranteed Token Redemption

We refer to the property where the IoT gateway has a valid payment token in return for the CO units offered to the IoT device by guaranteed token redemption. When handling such a token to the home cloud admin of the IoT, the home cloud admin verifies that the token is generated by one of its registered and subscribed IoT devices and pays back the hosting cloud admin, i.e., IoT_h has a subscription plan that can cover the required CO units. Also, the IoT gateway of CA_v can reclaim the charges of CO units it offered to IoT_h.

In the mutual authentication phase, IoT_h sends a pre-image value in its IoT hash chain as a payment token in addition to the Merkle proof of the tip to the IoT gateway of CA_v. Unsubscribed IoT devices cannot produce a valid Merkle proof in the CA_h Merkle tree. This is because of the second pre-image resistance of the hash function, i.e., given

$H(\mathcal{T}) = Y$, an adversary can find $\mathcal{T}'$ such that $H(\mathcal{T}') = Y$ with a negligible probability. Also, to check that the IoT subscription covers the amount of CO units required, the IoT gateway verifies that $H^n(X) = \mathcal{T}$.

In the redemption phase, CA_h pays back the charges of n CO units. The CA_h computes $\mathcal{S}_i = H(K||S_i)$ and checks the payment token in the IoT hash chain. Assuming that K is known only to CA_h, the IoT gateway has a negligible probability of finding a new pre-image value X' such that $H^{n'}(X') = \mathcal{T}_i$.

User Traceability

In case of any misbehaving IoT device, the IoT gateway can send the pseudo-identity S_i of the misbehaving IoT device to CA_h which can trace it back to its real identity.

Eavesdropping Attack

Information between the IoT device and the IoT gateway is encrypted under the session key K_s and any external adversary cannot eavesdrop on the exchanged information.

4.5 Performance Evaluation

In this section, we present the analysis of SKICAP concerning the communication overhead, and the computation and storage requirements. We summarize the performance evaluations in Table 5.2. We use the Raspbian 64-bit operating system on a 1.5 GHz 64-bit Quad-core ARM Cortex-A72 processor embedded in a Raspberry Pi 4 Model B/8GB. Then, we run the cryptographic operations shown in Table 4.3 for 1000 times to calculate the average computation time of each operation. For the simulation parameters, we consider SHA-256 over 1024-byte data for the Hash function. We also consider SHA-256 as the underlying message-digest algorithm for HMAC. For the symmetric key encryption,

we consider the AES-CBC mode with a key size of 128 bits. Since the time difference between the encryption and decryption is negligible, we assume the same time T_s for both operations. For the modular exponentiation, point multiplication, and bilinear pairing operations, the Python library bplib is used, which implements a bilinear pairing over a Barreto-Naehrig curve [1]. For calculating the PUF execution time, we consider a latch-based PUF embedded with an error-correcting code of a 256-bit response [152, 156].

4.5.1 Computation Overhead

For the IoT device, in the mutual authentication phase, it needs to perform $Log\ N$ random number generations besides 1 random number generation for the nonce $\mathcal{N}$. The IoT device performs $2 \times Log\ N$ hashes for generating the challenges for the PUF circuit. Moreover, for generating its temporary pseudo-identities, the IoT device does $2 \times Log\ N$ PUF calls and $Log\ N$ hashes plus 1 encryption operation for the nonce $\mathcal{N}$. Then, for computing the payment token, it performs $(L - n)$ hash-based computations and $Log\ N$ encryption operations over the Merkle proof and 2 encryption operations for the payment token and the tip. For deriving the session key, the IoT performs 1 hash-based computation. Thus, in total, the IoT device performs $2 \times Log\ N$ PUF calls, $1 + Log\ N$ random number generations, $3 \times Log\ N + (L - n) + 1$ hash-based computations and $(3 + Log\ N)$ encryption operations where n is the required CO units and N is the number of the IoT devices subscribed with CA_h.

The gateway performs at most $2 \times Log\ N$ keyed-hash computations to decode the temporary pseudo-identity of the IoT device, 1 hash-based computation for deriving the session key, and $3+Log\ N$ decryption operations for the nonce, payment token, the tip and the Merkle proof. Then, for verifying the payment token, the gateway performs n hash-based computations to verify the pre-image value and $Log\ N$ hash-based computations for verifying the Merkle proof. Thus, in total, the gateway performs at most $(1 + n\ + 3 \times Log\ N)$ hash-based computations and $3 + Log\ N$ decryption operations.

Table 4.2: Protocol storage, computation and communication requirements

Description	Value
IoT storage	$(1 + 1 \times Log\ N) \times 32$ bytes
Gateway storage	$(2 \times N_s + 3) \times 16$ bytes
IoT computation	$2 \times Log\ N$ PUFs, $1 + Log\ N$ RNG, $3 \times Log\ N + (L - n) + 1$ Hash, $(3 + Log\ N)$ Enc
Gateway computation	$(1 + n + Log\ N)$ Hash, $2 \times Log\ N$ HMAC, $3 + Log\ N$ Dec
Communication overhead	$113 + 48 \times Log\ N$ bytes

4.5.2 Execution Time

In our calculations for the delay imposed by SKICAP, we assume that the IoT device requires 10 CO units from the hosting cloud, out of 100 subscribed CO units. In addition, we neglect the bitwise XOR operations as it is a one-cycle operation. For more realistic results, we consider the number of registered IoTs with a home cloud admin in a country serving one application to be 2^{24} IoTs. This covers the expected number of deployed IoTs in applications such as healthcare, finance, industry and retail in Europe by the year 2023 as reported by the European Telecommunications Network Operators'(ETNO) association. To compare SKICAP with closely related works, the computation time imposed by our protocol on the IoT device and the IoT gateway is compared with [15, 29, 79, 109, 142] in Fig. 4.8. In the comparison, we consider the number of required cryptographic operations and the time required by each operation as reported in Table 4.3. One can see that SKICAP has the least total computation time. Note that, according to [118], the average number of the IoTs per person will increase up to 9 IoTs/user. Therefore, in LAMANCO [146], the IoT device user needs to plug in a smart card and remember a password for each IoT device, which will be non-practical with the increasing number of IoTs per user. It should also be noted that Nakkar et al. [106] does not achieve mutual authentication as it is designed to be a one-way authentication protocol for authenticating the server broadcasting messages.

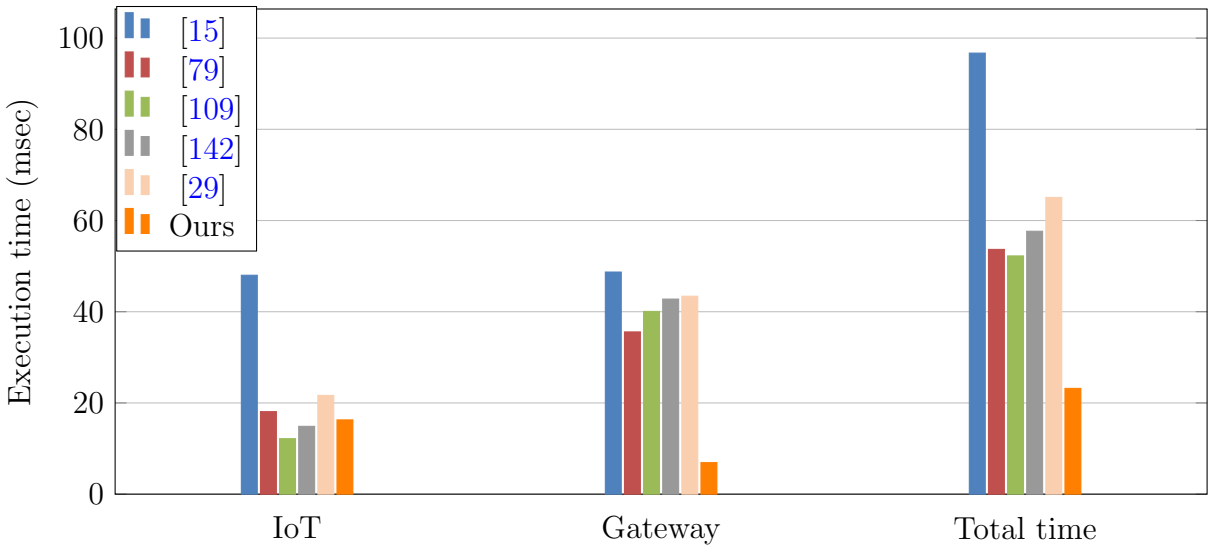

Figure 4.8: IoT device, IoT gateway and total execution time of [15], [79], [109], [142], [29] and SKICAP

Table 4.3: Execution time of various cryptographic operations

Description	Notation	IoT (msec)
Hash operation [1]	T_h	0.0487
HMAC operation [2]	T_{MAC}	0.0834
Symmetric Encryption/Decryption [3]	T_s	0.0446
RNG	T_{RNG}	0.0552
PUF call [4]	T_P	0.12
Bilinear Pairing operation [5]	T_b	15.33
Multiplication Point Operation [5]	T_M	2.7551
Modular Exponential Operation [5]	T_e	3.609

[1]: We consider SHA-256 over 1024 bytes data
[2]: We consider SHA-256 as the underlying message-digest algorithm
[3]: We consider AES-CBC mode with a key size of 128 bits.
[4]: We consider a latch-based PUF embedded with error-correcting code of a 256-bit response [152, 156].
[5]: We use the Python library bplib, which implements a bilinear pairing over a Barreto-Naehrig curve [1].

4.5.3 Communication Overhead

For our analysis, to maintain a security level of 128 bits, we assume the nonce $\mathcal{N}$, the challenge r and all the secrets to be 128 bits in length. We assume n to be 1 byte. Also, we assume the authenticated encryption tag to be 256 bits and the message to be of an arbitrary length. Thus, the total communication overhead is 16 bytes challenge

from the IoT gateway, followed by $32 \times Log\ N$ bytes for the temporary pseudo-identity of the IoT device plus the 32 bytes for the encryption of $\mathcal{N}$ and its increment. This is in addition to 1-byte which represents the required n units, $16 \times Log\ N$ bytes Merkle proof, 16 bytes payment token, and 16 bytes the tip of the hash chain. In the message exchange phase, the IoT device and the IoT gateway add a 32 bytes tag. Thus, the communication overhead sums up to $113 + 48 \times Log\ N$ bytes.

4.5.4 Storage Overhead

The IoT device needs to store a 16-byte seed of the hash chain. For PUF challenges, the IoT device stores the challenge C_1 of 128 bits and derives other C_i from it. Moreover, it requires $Log\ N$ memory, each of 16 bytes, for storing M_i to compute $\mathcal{K}_i$. Also, the IoT device needs $16 \times Log\ N$ bytes for the Merkle proof of the tip of the hash chain. Thus, the IoT storage requirement is $(\ 1\ +\ 1\ \times Log\ N) \times 32$ bytes.

The IoT gateway requires 3×16 bytes for the secret key S, the key K_1 and the Merkle root. For the redemption of CO charges, the IoT gateway keeps both the payment tokens X and corresponding pseudo-identity S_i of the interacting IoT devices. Assume that the IoT gateway can serve up to N_s IoT devices at each redemption interval. Thus, the gateway needs a storage of $(2 \times N_s\ + 3\ ) \times 16$ bytes.

4.6 Summary

We presented an inter-cloud mutual authentication protocol. To thwart the physical attacks on the IoT devices, the authentication keys are not stored on the IoT device, instead, they are computed using responses of a PUF circuit embedded on the IoT. To mitigate impersonation and replay attacks, the temporary pseudo-identity of the IoT device is composed of chained authenticated messages. We also provided formal security proofs for the design goals of our enhanced protocol. Additionally, we evaluated the

performance of the protocol concerning the communication overhead, and storage and computation requirements.

In this version of SKICAP, we consider a fixed and common tariff among the cloud admins and no price negotiation between IoT devices and the available hosting cloud admins. For future works, we plan to explore the integration of SKICAP with blockchain for price negotiation and charge redemption. A deployed smart contract can guarantee the redemption of charges from the home cloud admin in response to the computation offloading services provided by the hosting cloud admin. Moreover, the deployed smart contract can be used for price negotiation where an IoT device can get the best and lowest offered CO services.

Chapter 5

Mutual Authentication Privacy-Preserving Protocol with Forward Secrecy for the IoT-edge-cloud

5.1 Introduction

Anonymity can be achieved through anonymous authenticated schemes such as ring/group signature schemes [30, 116]. In such schemes, both the IoT device and the edge node should have access to public keys of registered IoT devices in the anonymity set for signature generation and verification which requires either extra communication between IoT devices or storage requirements. Moreover, registration of a new IoT device to the system requires updating all IoT devices and edge nodes with the public key of the new IoT device which further affects the system's scalability.

To address the aforementioned problems in anonymous authenticated schemes, Certificate-Less Public-Key Cryptography (CL-PKC) authentication and key agreement

schemes have been considered where the generation of the private keys and public keys are split between the IoT device and the Key Generation Center (KGC) [7]. In [92], Li *et al.* proposed an Elliptic Curve Cryptography (ECC)-based mutual authentication and key exchange protocol for IoTs. The proposed protocol achieves mutual authentication and forward secrecy property. The mutual authentication property ensures the legitimacy of the transacting entities while the forward secrecy property ensures the security of the past session keys in case of compromising long-term secrets. However, Li's protocol failed to maintain the privacy of the IoTs since the communicating entities have to send their identities to complete the authentication process. Similarly, Ying *et al.* [157] proposed an ECC-based mutual authentication and key establishment protocol in 5G networks. The proposed protocol achieves mutual authentication and forward secrecy. Regarding anonymity, the protocol obfuscates the real identity of the IoT device. However, it suffers from linkability where an external adversary can link the authentication requests of the IoT device since it has the same pseudo-anonymous identity.

Consider a scenario where a malicious network admin monitors the deployed edge nodes in a typical e-healthcare system [57]. By linking the requests from the embedded sensors on the patient's body, a malicious network admin can profile and trace the patient during their visit to the hospital, the doctor clinics, and medical laboratories. In this chapter, we consider the anonymity of the sender and the unlinkability of the authentication request for the serving edge node.

Motivated by the aforementioned protocols, in this chapter, we aim to achieve the following goals: 1) design a privacy-preserving mutual authentication protocol for the IoT-edge-cloud paradigm where an external adversary or edge node, controlled by a malicious network admin, cannot efficiently identify, or co-relate the incoming requests, and 2) enable scalability of the protocol without the overhead of updating all other IoT devices and edge nodes with public keys of new IoT devices.

We propose MAPFS, a Mutual Authentication Privacy-preserving protocol with

Forward Secrecy for the IoT-edge-cloud. MAPFS is resilient to replay attacks; also, it achieves forward and backward secrecy properties and it ensures unlinkability between IoT requests concerning the edge node. In MAPFS, the IoT device sends an authentication token to a nearby edge node to prove its legitimacy. To achieve unlinkability between the authentication tokens of the same IoT device, the IoT device randomizes such authentication tokens. Such a randomization process is essential as it provides the unlinkability between the authentication requests and prevents the external adversaries and the edge nodes from correlating the authentication requests and profiling the IoT device. Table 5.1 provides a comparison of the security properties of our proposed protocol, MAPFS, against other related ones. In our comparison, we consider the mutual authentication, the anonymity of the sender and the unlinkability of the IoT requests from external adversaries and edge nodes. Additionally, we consider other functional properties, such as scalability, where adding an IoT device is easy by issuing an authentication token to the new device without requiring protocol re-initialization or re-registration of other IoT devices. Our protocol, MAPFS, achieves mutual authentication, sender anonymity and session unlinkability. Note that PUF-based protocols [92, 123, 127] require the presence of a PUF circuit for running the protocol. If the PUF circuit is not present, the protocol cannot be executed, and it will lose its hardware compromise resilience security property.

Table 5.1: Comparison with other related protocols.

Security/Functionality Properties	[107]	[92]	[157]	[6]	[56]	[130]	[84]	[141]	[127]	[123]	[80]	[81]	[94]	[82]	[32]	MAPFS
Mutual Authentication	×	✓	✓	✓	×	✓	×	✓	✓	✓	✓	✓	✓	✓	×	✓
Anonymity w.r.t External adversary	×	×	✓	✓	×	✓	×	×	✓	✓	✓	✓	✓	✓	×	✓
Anonymity w.r.t edge node	×	×	✓	×	×	×	×	×	✓	✓	×	✓	✓	✓	×	✓
Unlinkability w.r.t External adversary	×	×	×	✓	×	✓	×	×	✓	✓	✓	✓	✓	✓	×	✓
Unlinkability w.r.t edge node	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	✓
Bilinear-Pairing Free	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	×	×	×	×	×	✓
Extra Hardware-Free	✓	×	✓	✓	✓	✓	×	×	×	×	✓	✓	✓	✓	✓	✓
Scalability	✓	✓	✓	×	×	×	✓	✓	×	×	✓	✓	✓	✓	✓	✓
IoT-Edge-Cloud compatibility	×	✓	✓	×	✓	×	✓	✓	✓	✓	✓	✓	✓	×	✓	✓

5.2 System Model and Threat Model

In what follows, we present our system model and threat model.

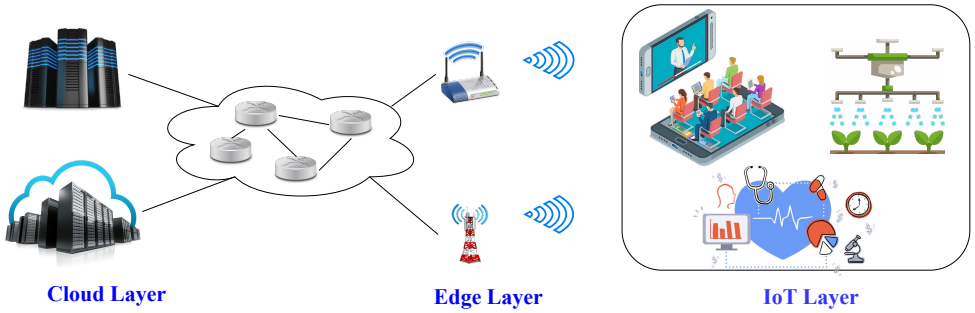

Figure 5.1: MAPFS system model

5.2.1 System Model

We adopt the IoT-edge-cloud paradigm as illustrated in Fig. 5.1. Our system model is composed of the IoT layer, the edge layer, and the cloud layer. Edge nodes are connected to the cloud layer through the core network. The role of each layer is detailed as follows:

1. IoT Layer: Such layer is composed of the sensor/IoT devices, which are limited-resource devices with internet connectivity and enrolled in many applications (e.g., transportation, healthcare, virtual reality, $\cdots$, etc.). IoT devices sense and gather the needed information and send most of the data to the edge layer to benefit from the computation power of the IoT gateway to meet the expected QoS requirements.

2. Edge Layer: An intermediate layer between the IoT and cloud layers. It brings a part of the cloud computing infrastructure closer to the end-users in the form of IoT gateways to improve the latency of real-time applications. The computation-capable edge nodes can be a router, a switch, or a proxy server [75]. It offers computation offloading services to limited-resource IoT devices.

3. Cloud Layer: A top layer that provides scalable computing resources, storage, and services that complement edge computing. It enables data storage, complex data analytics, machine learning, and other resource-intensive tasks that cannot be efficiently performed at the edge nodes.

Moreover, in our protocol, we make use of a Trusted Third Party (TTP) which is

the KGC that is responsible for initializing the system parameters and issuing the signing keys for IoT devices and IoT gateways, in the registration phase.

5.2.2 Threat Model

We adopt the CK adversary model defined in 3.2.2. Based on the information revealed to $\mathcal{A}$, we define the adversary attacks as follows.

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

5.3 Protocol Design

Our protocol, MAPFS, makes use of two-party ECDH key exchange [42], Schnorr Zero-Knowledge Proof of discrete log knowledge [121], and Schnorr signature [120]. The Diffie-Hellman protocol enables the IoT device and the IoT gateway to establish the session key. However, the ECDH is an unauthenticated key agreement protocol. Therefore, the Schnorr signature is used to authenticate the IoT gateway to the IoT device, and vice versa. The IoT gateway uses a signing key, issued by the KGC, to sign its authentication message for the IoT device. Similarly, the IoT device uses a signing key to sign its authentication message. To maintain the unlinkability of the IoT requests, the protocol randomizes the authentication message of the IoT device by multiplying it with a random number. Furthermore, the Schnorr ZKP is used to prove the knowledge of the random number that relates the randomized authentication token to our public system parameters.

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

follows.

5.3.1 Setup Phase

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

5.3.2 Registration Phase

We assume that the registration phase runs in a secure environment. Figure 6.3 illustrates the registration phase between the gateway$_w$ and the KGC. The IoT gateway randomly selects $x_w \xleftarrow{r} Z_q^*$ as its partial private key and computes the partial public key $X_w = x_w P$. Then, the IoT gateway sends its identity ID_w along with the partial public key X_w to the KGC. After that, the KGC randomly selects $y_w \xleftarrow{r} Z_q^*$ and computes the partial public key $Y_w = y_w P$ (i.e., the two partial keys X_w, Y_w are generated by the IoT gateway and the KGC, respectively, and serve as the public key of the gateway). Then, the KGC computes $h_w = H_1(ID_w||X_w||Pub_{gc}||Y_w)$ to bind the public parameters ID_w, X_w, Pub_{gc}, Y_w, together. Then, the KGC issues the signing key $\sigma_w = s_{gc} + h_w y_w$ which is verified on the gateway by validating $\sigma_w P \stackrel{?}{=} Pub_{gc} + H_1(ID_w||X_w|||Pub_{gc}||Y_w)Y_w$. Note that, the KGC includes the secret s_{gc} to indicate the issuing of the signing key by the KGC and binds the h_w value with the partial private key of the gateway y_w to indicate

that this signing key is issued to the IoT gateway with public key Y_w.

Similar to the registration of the IoT gateway, the KGC includes s_{gc}, $h_a y_a$ in the IoT signing key to indicate the issuing of such signing key by the KGC. Moreover, the KGC relates the h_a with the partial private key of the IoT device y_a to indicate that this signing key is issued to the IoT device with public key Y_a as shown in Fig. 5.3. However, since our protocol is privacy-preserving and the IoT authentication request should be unlinkable, the registration phase of the IoT device is a little bit different from the gateway registration. The KGC computes the $h_a = H_2(ID_a||X_a|||Pub_{gc}||Y_a||h_x)$ to bind the IoT identity ID_a, the IoT public key $< X_a, Y_a >$, and the KGC public key Pub_{gc}, together. Moreover, the KGC includes the secret key h_x in the computation of h_a to prevent the exhaustive searching by an external adversary and identifying the IoT device identity through hashing the public parameters ID_a, $< X_a, Y_a >$, Pub_{gc}, in the IoT authentication request, if h_x is not included. Furthermore, the KGC includes the term y_a in the signing key of the anonymous IoT device to provide a Zero-knowledge proof of the knowledge of the discrete log, h_a, of the randomized points P_3 to the base point P_1. Without a valid value of h_a that links $P_3 = h_a P_1$, the KGC cannot revoke the identity of a misbehaving IoT device.

To enhance the scalability and alleviate the bottleneck problem during the registration of IoT devices, we allowed an IoT device to send its registration data $< ID_a||X_a||h_x >$ and a symmetric key s_k to the KGC through deployed gateways (i.e., note that, in this case, the public key of the KGC Pub_{gc} is assumed to be embedded on the IoT device). Afterward, the KGC decrypts the incoming data and generates the partial public key Y_a for the IoT device along with the signing key σ_a and the $h_a = H_2(ID_a||X_a|||Pub_{gc}||Y_a||h_x)$. Then, it does symmetric encryption using the secret s_k for $< ID_a||X_a||\sigma_a||Y_a||h_a >$ and sends the encrypted data to the IoT device through edge nodes along with the integrity term $M_1 = H(ID_a||X_a||\sigma_a||Y_a||h_a)$. The IoT device decrypts the incoming message and verifies $M_1 \stackrel{?}{=} H(ID_a||X_a||\sigma_a||Y_a||h_a)$ and $\sigma_a P \stackrel{?}{=} Pub_{gc} + h_a Y_a + Y_a$. Then, the IoT device

stores σ_a, x_a, and h_a in its memory. This solution enhances the scalability and alleviates the bottleneck problem during the registration of IoT devices but it requires the public key of the key generation center to be embedded in the IoT device. Another alternative to enhance the scalability and alleviate the bottleneck problem is to utilize a hierarchical structure comprising a root KGC and sub-local KGCs [150]. This architecture enables IoT devices to register with their respective sub-local KGCs, providing a more distributed and efficient approach.

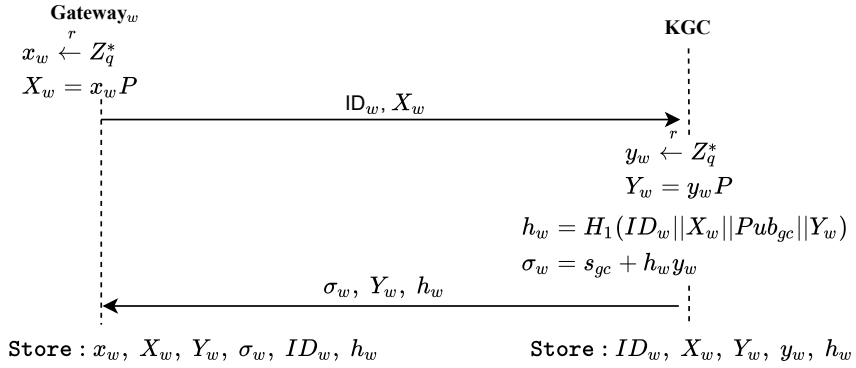

Figure 5.2: The IoT gateway registration

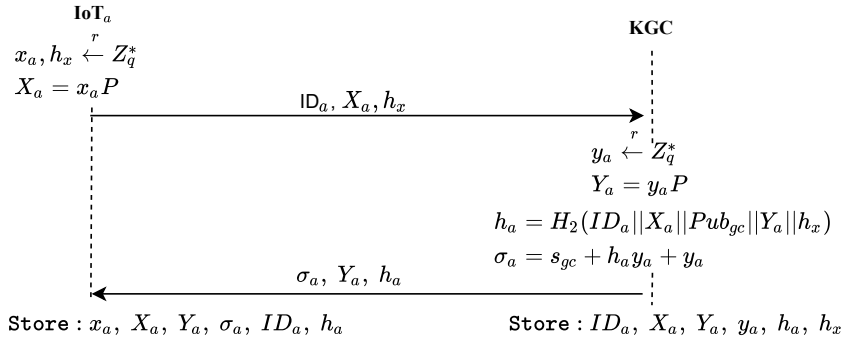

Figure 5.3: The IoT device registration

5.3.3 Mutual Authentication and Key Agreement Phase

Consider a session i between IoT_a and gateway_w, as shown in Fig. 5.4, the IoT_a randomly generates the random nonces $r_1,\ r_2,\ r_3,\ r_4$ where r_1 is used in deriving the IoT one-time public key $A = r_1 X_a$, r_2 is used in randomizing the IoT signing key, σ_a while r_3 and r_4 are used in generating the ZKP commitments T_1 and T_2. The IoT sends Hello message along with the IoT one-time public-key A to the IoT gateway. Then, the IoT gateway generates r_5 and computes the one-time public key $W = r_5 X_w$. Afterwards, it computes I_g and the gateway signature $\sigma_z = I_g \sigma_w + r_5 x_w$ such that $I_g = H_3(A||W)$. Such σ_z will be verified by the IoT device by checking $\sigma_z P \stackrel{?}{=} I_g Pub_{gc} + I_g H_1(ID_w||X_w||Pub_{gc}||Y_w)Y_w + W$. Upon receiving the IoT gateway message, $< W, ID_w, X_w, Y_w, \sigma_z >$, the IoT device verifies the received σ_z. Upon the successful verification of the received σ_z and authenticating the IoT gateway, the IoT device computes the session key $K_s = H_0(r_1 x_a W)$; otherwise, it terminates the session with such IoT gateway. After that, it computes the instantaneous base point $P_1 = r_2 Y_a$, and randomized points $P_2 = r_2 Pub_{gc}$ and $P_3 = r_2 h_a Y_a$. For the ZKP, the IoT device computes the two commitments $T_1 = r_3 Pub_{gc}$ and $T_2 = r_4 P_1$ to prove the knowledge of r_2 (resp. h_a) in the statement $\exists\ r_2\ s.t.\ P_2 = r_2 Pub_{gc}$ (resp. $\exists\ h_a\ s.t.\ P_3 = h_a P_1$). After that, the IoT device computes $I_a = H_4(A||P_1||P_2||P_3||T_1||T_2||W)$. Then, the IoT computes the randomized signature $\sigma_t = I_a r_2 \sigma_a + r_1 x_a$ that will be verified by the IoT gateway by checking $\sigma_t P \stackrel{?}{=} I_a(P_1 + P_2 + P_3) + A$. Also, the IoT computes the ZKP responses $s_1 = r_2 I_a + r_3,\ s_2 = h_a I_a + r_4$. Later on, the IoT gateway verifies these ZKP responses by checking $s_1 Pub_{gc} \stackrel{?}{=} I_a P_2 + T_1$ and $s_2 P_1 \stackrel{?}{=} I_a P_3 + T_2$. The IoT device sends points $P_1,\ P_2,\ P_3$ along with the commitments $T_1,\ T_2$, IoT signature σ_t and ZKP response $s_1,\ s_2$ to the IoT gateway. Then, the IoT gateway verifies the IoT signature σ_t and the ZKP responses $s_1,\ s_2$ for authenticating the IoT device.

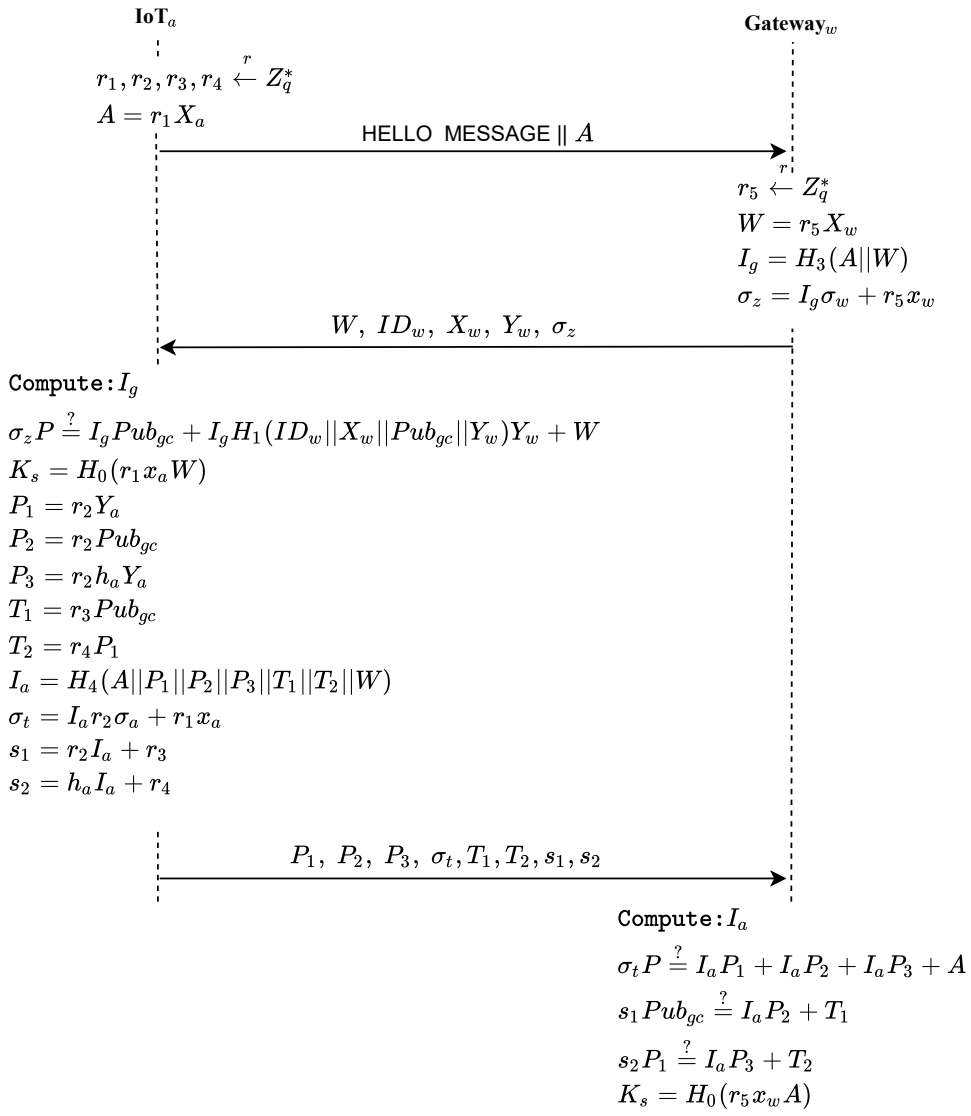

Figure 5.4: Mutual authentication phase

5.3.4 Revoking the anonymity of misbehaving IoTs

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

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

5.4 Security Analysis

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

5.4.1 Adversary Model

In our proposed protocol, we define the session identifier $\mathcal{S} = H_3(A||W)$ as the hash value of the IoT device one-time public key A and the gateway one-time public key W. Moreover, the protocol participants, IoT_a and IoT gateway_w, are said to be partners if the following conditions are met: 1) the two participants are in the accept state, 2) they have the same session identifier $\mathcal{S}_i = < A||W >$ and 3) the partner identifier of IoT_a is IoT gateway_w and vice versa [78]. We model the adversary capabilities in interacting with the protocol participant $\mathcal{O}$ (i.e., IoT device or the IoT gateway) by the following queries.

- **H( ):** The hash function is simulated as a random oracle. For each simulation H_i, a list $\mathcal{L}_i$ is maintained to keep the input in_i and the output out_i. When queried by $\mathcal{A}$, if the input in_i is found in the stored list $\mathcal{L}_i$, the output out_i is returned; otherwise, a random string $out_i \xleftarrow{r} \{0,\ 1\}^{l_i}$ is returned where l_i is the output length of the hash function H_i and the entry in_i and out_i is added to the stored list $\mathcal{L}_i$.

- **Send($\mathcal{S}, m, \mathcal{O}$):** It is used to model the adversary's active attacks on the system, such as replay attacks, impersonation attacks, and injection attacks. It allows the

adversary to act as a legitimate entity and send a message m to the protocol participant $\mathcal{O}$ in the session $\mathcal{S}$. It responds according to the protocol specifications, which depend on its role and current internal state.

- **Execute(IoT_a, G_w):** It is used to model the adversary's passive attacks by generating the transcript $\mathcal{T}$ of the transmitted messages during an honest protocol execution between the protocol participants, namely the IoT device IoT_a and the IoT gateway G_w.

- **SSReveal($\mathcal{O}, \mathcal{S}$):** This query allows $\mathcal{A}$ to obtain the internal state of the protocol participant $\mathcal{O}$ during the execution of the protocol in session $\mathcal{S}$, including any relevant internal variables such as r_1, r_2, r_3, r_4 for the IoT device and r_5 for the IoT gateway.

- **SKReveal($\mathcal{O}, \mathcal{S}$):** This query allows $\mathcal{A}$ to determine the session key (i.e., K_s in our protocol) held by the participant $\mathcal{O}$ during the session $\mathcal{S}$.

- **Corrupt($\mathcal{O}$):** This query allows $\mathcal{A}$ to obtain the long-term secrets used by the participant $\mathcal{O}$ in the protocol, such as x_a, σ_a, h_a for the IoT device and x_w, σ_w for the IoT gateway.

Mutual authentication. Intuitively, we say that MAPFS ensures mutual authentication, if it is infeasible for $\mathcal{A}$ to impersonate an IoT device to an honest gateway, and it is also infeasible for $\mathcal{A}$ to impersonate a gateway to an honest IoT device. The Mutual Authentication (MA) security of a MAPFS protocol supporting n IoT devices and m gateways is modeled by an experiment $Auth$, where $\mathcal{A}$ attempting to impersonate IoT_j (resp. gateway G_j) is allowed to invoke the **Send** query with $\{IoT_1, IoT_2, \ldots, IoT_n\}$ - $\{IoT_a\}$ (resp. $\{G_1, G_2, \ldots, G_m\}$ - $\{G_j\}$). At the end, $\mathcal{A}$ wins if it outputs a valid login message for the target IoT device IoT_a or the target IoT gateway G_w. The advantage of $\mathcal{A}$ is denoted by $Adv^{Auth}_{\mathsf{MAPFS}}(\mathcal{A})$.

The session key K_s of $\mathcal{O}$ is said to be fresh if the following conditions hold: 1) no **SKReveal** has been invoked on $\mathcal{O}$ or its partner, and 2) at most one corrupt query, either **SSReveal** or **Corrupt** has been invoked by $\mathcal{A}$ on $\mathcal{O}$ or its partner. It is reasonable that if $\mathcal{A}$ gets the secret parameters of both the protocol participants, $\mathcal{A}$ can compute the session key [78, 92].

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

5.4.2 Security Analysis

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

Theorem 5.1 *Under the assumption of the intractability of ECDLP, MAPFS is MA-secure where for any PPT adversary $\mathcal{A}$, $Adv^{Auth}_{MAPFS}(\mathcal{A}) \leq \epsilon$.*

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

In the initialization phase, $\mathcal{B}$ initializes the protocol and sets the public key $Pub_{gc} = Q$. Also, it sets the IoT device IoT_a as the target device for $\mathcal{A}$ (i.e., it is required to

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

In the training phase, $\mathcal{A}$ performs the **Send** query to send the tuple < W, ID_w, X_w, Y_w > to the protocol participants $\mathbb{S}_I = \{IoT_1,\ IoT_2,\ \ldots,\ IoT_n\}$ to obtain the IoTs authentication tuple < $P_1,\ P_2,\ P_3,\ \sigma_t,\ T_1,\ T_2,\ s_1,\ s_2$ >. Note that, the target IoT device $IoT_a \notin \mathbb{S}_I$.

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

$$\begin{aligned} \sigma_t &= I_a r_2(s_{gc} + y_a h_a + y_a) + r_1 x_a \\ s_1 &= r_2 I_a + r_3 \\ s_2 &= h_a I_a + r_4 \end{aligned} \tag{5.1}$$

where $I_a = H_4(A||P_1||P_2||P_3||T_1||T_2||W)$. According to the forking lemma [113, 114], $\mathcal{B}$ and $\mathcal{A}$ can repeat the above game with the same random nonces $r_1,\ r_2,\ r_3,\ r_4$ and a different hash oracles $\mathcal{H}$ until $\mathcal{A}$ outputs another valid login message < $P_1,\ P_2,\ P_3,\ \sigma_t',\ T_1,\ T_2,\ s_1',\ s_2'$ > such that

$$\begin{aligned} \sigma_t{}' &= I_a{}' r_2(s_{gc} + y_a h_a + y_a) + r_1 x_a \\ s_1{}' &= r_2 I_a{}' + r_3 \\ s_2{}' &= h_a I_a{}' + r_4 \end{aligned} \tag{5.2}$$

from (1) and (2),

$$
\begin{aligned}
r_2 &= (I_a - I_a{}')^{-1}(s_1 - s_1') \\
h_a &= (I_a - I_a{}')^{-1}(s_2 - s_2') \\
(I_a - I_a{}')^{-1}(\sigma_t - \sigma_t{}') &= r_2 s_{gc} + r_2 h_a y_a + r_2 y_a
\end{aligned}
\tag{5.3}
$$

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

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

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

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

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

to impersonate the target IoT gateway w. Then, the following condition holds

$$\sigma_z = I_g(s_{gc} + y_w h_w) + r_5 x_w \tag{5.4}$$

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

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

$$\sigma_z{}' = I_g'(s_{gc} + y_w h_w) + r_5 x_w \tag{5.5}$$

from (5.4) and (5.5),

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

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

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

Theorem 5.2 *Under the intractability assumption of ECDDH, MAPFS is SS-secure where for any PPT adversary $\mathcal{A}$, $Adv^{SSec}_{MAPFS}(\mathcal{A}) \leq \epsilon$.*

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

the following game between $\mathcal{A}$ and $\mathcal{B}$.

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

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

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

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

- $\mathcal{A}$ invokes **Corrupt** on either the IoT device IoT_a or the IoT gateway G_w. Moreover, $\mathcal{A}$ invokes **SSReveal** on the protocol instance π_i.

- $\mathcal{A}$ invokes the **SKReveal** query on the protocol instance π_i to get the computed session key K_s.

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

message $\mathcal{M}_2$ from the IoT gateway to the IoT device in the protocol transcript $\mathcal{T}$. After that, the IoT device adds $< A, P_1, P_2, P_3, T_1, T_2, W >$ to $\mathcal{L}_4$ as input to the hash query with output I_a. Then, it outputs $< P_1, P_2, P_3, \sigma_t, T_1, T_2, s_1, s_2 >$ as the third message $\mathcal{M}_3$ from the IoT device to the IoT gateway in the protocol transcript $\mathcal{T}$.

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

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

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

MAPFS Freshness

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

In our protocol, the IoT device starts the session with Hello message along with a fresh one-time IoT public key $A = r_1 X_a$. The IoT gateway replies with a fresh one-time

gateway public key $W = r_5 X_w$. These fresh A, W are used in the computation of the integrity terms $I_g = H_3(A||W), I_a = H_4(A||P_1||P_2||P_3||T_1||T_2||W)$ which are embedded in the gateway signature $\sigma_z = I_g \sigma_w + r_5 x_w$ and the IoT signature $\sigma_t = I_a r_2 \sigma_a + r_1 x_a$.

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

Unlinkability of the IoT requests

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

Perfect Forward Secrecy

This property is maintained if the compromise of the long-term key or the current session key does not lead to the leakage of the past session keys [36]. Here, in our protocol, the session key $K_s = H_0(r_5 x_w A) = H_0(r_1 x_a W) = H_0(r_1 r_5 x_a x_w P)$ where r_1 and r_5 are two random nonces generated by the IoT device and the IoT gateway, respectively. Therefore,

the protocol is said to achieve perfect forward secrecy since the computation of the session key depends on the long-term key $x_a,\ x_w$ of the IoT device and the IoT gateway as well as the fresh randoms $r_1,\ r_5$ generated by the IoT device and the IoT gateway during the new session.

Backward Secrecy

This property is maintained when an adversary who has access to the protocol state values (i.e., $r_1,\ r_2,\ r_3,\ r_4,\ A,\ P_1,\ P_2,\ P_3,\ r_5,\ W$) cannot compute the previous session keys [88]. The computation of the session key of the session i depends on the randoms generated by the IoT gateway and the IoT device during the session i where $K_s = H_0(r_1 r_5 x_a x_w P)$. Therefore, compromising the IoT device state value during session i does not leak any information about the session key of the sessions $i-1,\ i-2,\ \cdots,\ 2,\ 1$. Hence, MAPFS achieves the backward secrecy property.

5.5 Performance Evaluation

It is important to consider the efficiency of the proposed protocol by analyzing its performance in terms of the i) communication overhead which consists of messages exchanged between the communicating entities before the actual transfer of information, i.e., these are the messages exchanged between the IoT device and IoT gateway to achieve the mutual authentication and session key establishment, ii) storage requirement on the IoT device and the IoT gateway, i.e., the secrets stored on the IoT device and the IoT gateway to achieve mutual authentication and session key derivation, and iii) computation cost which involves the operations that are done by the IoT device and the IoT gateway during the authentication process and session establishment. In our performance analysis, we assume 128-bit random values and 128-bit ID. Also, we assume a 256-bit elliptic curve which typically provides nearly a 128-bit security level [5]. To perform the hash functions

H_0, H_1, H_2, H_3, H_4 which incur EC points in their domains, we use the x and y coordinates for the representation of the EC points. Moreover, as the codomains for the hash function are in Z_q, we perform modular q operation on the output of the hash function where q is a 256-bit. The summary of our performance analysis is presented in Table 5.2.

Table 5.2: Summary of our performance analysis

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

Table 5.3: Average cryptographic overhead time using 1000 runs

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

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

5.5.1 Storage Requirements

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

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

5.5.2 Computation Cost

In the authentication process, the IoT device generates the random nonces $r_1,\ r_2,\ r_3,\ r_4$ and does 6 scalar point multiplications to compute the randomized points $A,\ P_1,\ P_2,\ P_3$ and the commitments $T_1,\ T_2$. For computing σ_t, the IoT device does 1 hash operation to compute the I_a. For verifying the authentication token of the IoT gateway, the IoT device does 2 hash operations, 3 scalar point multiplications, and 2 point additions. Additionally, the IoT device does 1 hash operation and 1 scalar multiplication for computing the session key.

On the other side, the IoT gateway generates the random nonce r_5 and performs 1 scalar point multiplication to compute W. Moreover, the IoT gateway does 8 scalar point multiplications and 4 point additions to verify the authentication request of the IoT device. For computing the session key, the IoT gateway does 1 scalar multiplication. During the authentication process, the IoT gateway does 3 hash operations to compute the $I_a,\ I_g$ and session key.

5.5.3 Communication Overhead

The IoT device initiates the authentication process by sending Hello message along with a 2×128-bit randomized EC point A. The IoT gateway responds with a 9×128-bit

Table 5.4: Performance comparison based on the computation complexity

Protocol	Operations on IoT and Gateway Sides	Total Time (msec)
[84]	4 T_e + T_b	15.93
[80]	9 T_{sm} + 4 T_{pa} + 1 T_e+ 1 T_b + 10 T_h	17.737
[81]	5 T_{sm} + 3 T_{pa} + 1 T_e+ 1 T_b + 11 T_h	16.1627
[94]	7 T_{sm} + 1 T_{pa} + 1 T_e+ 1 T_b + 12 T_h + 2 T_s	17.1
[82]	T_e + 6 T_{sm} + 3 T_b	40.44
[32]	4 T_{sm} + T_e + 1 T_b	14.8
MAPFS	5 T_{rng} + 6 T_{pa} + 20 T_{sm} + 7 T_h	8.8899

message $< W,\ ID_w,\ X_w,\ Y_w,\ \sigma_z >$. In turn, the IoT device replys with 16 $\times$ 128-bit message $< P_1, P_2, P_3, \sigma_t, T_1, T_2, s_1, s_2 >$. Thus, in total, the communication overhead between the IoT device and the IoT gateway is 27 $\times$ 128 bits (i.e., 432 bytes).

Compared with other protocols in [56], [84], [80], [81], [94], [82], [32], MAPFS has the highest communication overhead. However, protocols in [56], [84], and [32] which require 96, 64, 128 bytes, respectively, do not offer mutual authentication. Meanwhile, protocols in [80–82,94] require 192, 192, 352, and 240 bytes respectively, but fail to ensure the IoT unlinkability from the serving IoT gateway, as seen in Table 5.1. This creates a vulnerability that allows an external attacker to compromise the IoT gateway and profile the IoT device.

5.5.4 Execution Time

We show the total computation-overhead time of the used cryptographic primitives in our protocol MAPFS compared to other related protocols, [84], [80], [81], [94], [82], [32] in Fig. 6.5. MAPFS has the lowest computation time compared to other protocols and it provides the unlinkability of the IoT request to the IoT gateway beside other security properties.

In our comparison, we consider the number of the required cryptographic operations on both the IoT device and gateway sides as reported in Table 6.3 and the average time required by each operation as shown in Table 6.2. Note that, we neglect modular

operations (i.e., multiplication and addition) as they require microsecond execution time. For measuring the average execution time, we use a Raspberry Pi 4 Model B/8GB embedded with a 1.5 GHz 64-bit Quad-core ARM Cortex-A72 processor running the Raspbian 64-bit operating system. We run the cryptographic primitives for 1000 times to compute the average execution time. For more resource-constrained IoT devices that are beyond the Raspberry Pi capabilities, the Arm Cortex M0 48 MHz ATECC508A HW accelerated as in [2, 106] can be considered. It offers 0.113 msec AES timing, 0.361 msec Hash timing, 0.722 msec HMAC timing, and 2 msec random number generator timing.

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

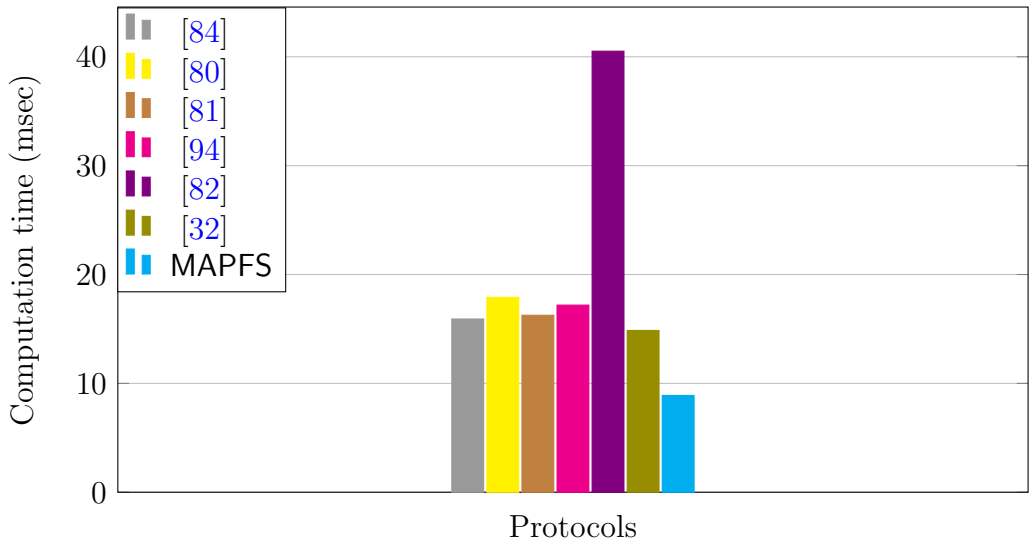

Figure 5.5: Evaluations of the overhead associated with cryptographic primitives in protocols [84], [80], [81], [94], [82], [32] and MAPFS

5.6 Summary

In this chapter, we proposed MAPFS, a privacy-preserving mutual authentication protocol between the IoT device and the IoT gateway for computation offloading services in the IoT-edge-cloud paradigm. MAPFS achieves anonymity of the IoT device, session unlinkability, and perfect forward secrecy for the established session key with revocation ability for the misbehaving IoT device. Our protocol distinguishes itself from certificate-based anonymous authenticated protocols in that anonymity can be achieved without additional communication or storage overheads.

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

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

Chapter 6

Conditional Privacy-preserving Message Authentication Protocol for VANET Emergency Message Exchange

6.1 Introduction

Vehicular Ad Hoc Network (VANET) still faces problems related to message authentication and privacy since such vehicular communication is carried out over open wireless channels [46]. An external adversary can collect sensitive messages. Furthermore, such an adversary can tamper with, forge, or replay traffic information and spread it through VANETs to cause serious damage (e.g., traffic chaos or accidents). Therefore, VANETs require secure and efficient vehicular communication protocols that guarantee message authentication and integrity [74]. Moreover, the privacy of the vehicle should be maintained, where an external adversary may try to obtain information about the vehicle's current position, velocity, and direction. Conditional privacy-preserving is also

an important security aspect in VANETs, where a Trusted Authority (TA) traces back the sender's identity in case of misbehavior (e.g., false traffic information, false location information). Efficient and secure solutions such as [9, 131, 160] have been proposed for V2V communications. Most of the existing solutions are based on a central Public-Key Infrastructure (PKI), where a centralized Certificate Authority (CCA) is responsible for issuing identity certificates for vehicles to prove their legal identity [100].

This requires that vehicles in the same administrative domain are served by a single CCA and gives rise to a single point of failure. Consider the case where a traveling vehicle (i.e., relocating from one city to another city) reports a traffic incident, such as a car accident, traffic congestion, or a fire, to the TMA. In such cases, the vehicle transmits traffic-related messages to prompt necessary actions, such as rerouting incoming vehicles and ensuring prompt responses from the police, firefighters, and ambulances. Nevertheless, the TMA and vehicles do not share the same trust domain. In such cases, communication between the vehicle and the city TMA for reporting accidents, traffic congestion, or fires becomes challenging. The premise of authentication between communicating entities (i.e., vehicle and TMA) is that they trust the CCA that issues identity certificates for both entities [83]. Moreover, sending such traffic-related messages exposes the vehicle's location. This highlights the necessity for a cross-domain communication protocol, allowing vehicles from different Enterprise Certificate Authority (ECA) (i.e., many enterprises prefer to issue certificates to their vehicles themselves and establish their ECA [155]) domains to communicate with the city TMA while maintaining vehicle anonymity and ensuring message authentication.

In this chapter, we propose a conditional privacy-preserving cross-domain communication protocol. This protocol allows a vehicle, when traveling to another city or country, to obtain an authentication token from their ECA and send an authenticated traffic message about traffic conditions (i.e., car accidents, congestion, and fire) to the city's TMA without prior registration. The proposed protocol not only maintains the

anonymity of the sender vehicle but also ensures resilience to message replay and modification attacks. Furthermore, it enables the traceability of misbehaving vehicles through collaboration among distributed KGCs, revealing the identity of the misbehaving vehicle in case of any misconduct.

In this chapter, we introduce CP-MAVE, a Conditional Privacy-preserving protocol with Message Authentication designed for cross-domain communications in VANET for Emergency message exchange. This efficient and secure communication protocol enables traveling vehicles to communicate with the city's TMA, located in a different trust domain, without requiring any prior registration with the city's TMA. CP-MAVE ensures unforgeability against chosen message attacks and the anonymity of sender vehicles to the TMA. It also exhibits resilience against message modification, replay attacks, and impersonation attacks. In the event of misconduct, CP-MAVE guarantees the traceability of misbehaving vehicles, with distributed KGCs collaboratively tracing back the identity of the sender vehicle.

Table 6.1 provides a comparison of the security properties of our proposed protocol, against other vehicular communication protocols. In our comparison, we consider message authentication, message integrity, the confidentiality of the transmitted data, and the anonymity of the sender. Additionally, we consider other functional properties, such as the support of the proposed protocol for cross-domain communications, and the dispense of the proposed protocol for heavy bi-linear pairing operations. Our proposed protocol is a privacy-preserving cross-domain authentication protocol for vehicular communications.

Table 6.1: Comparison with other message authentication protocols for VANETs.

Security/Functionality Properties	[8]	[69]	[139]	[154]	[45]	[73]	[49]	[147]	[149]	[110]	[144]	[145]	[81]	[140]	[50]	[153]	[17]	CP-MAVE
Message Authentication	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Message Integrity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Privacy-Preserving	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓
Cross-Domain Communication	×	×	×	✓	✓	✓	✓	×	×	×	×	×	×	×	✓	✓	×	✓
Bilinear-Pairing Free	×	×	✓	✓	✓	✓	✓	×	✓	✓	✓	×	×	×	×	×	×	✓
Traceability	✓	✓	✓	×	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

6.2 System Model and Threat Model

In what follows, we present our system model and threat model.

6.2.1 System Model

We adopt the architecture illustrated in Fig. 6.1. Our system model comprises Traffic Management Agencies (TMAs), Enterprise Certificate Authorities (ECAs), Road Side Units (RSUs), and On-Board Units (OBUs). Vehicle's OBUs communicate with RSUs through DSRC protocol, while communications between RSUs and the city's TMA occur through secure wired channels. Moreover, there are distributed KGCs, which collaboratively trace back the identity of the misbehaved vehicle.

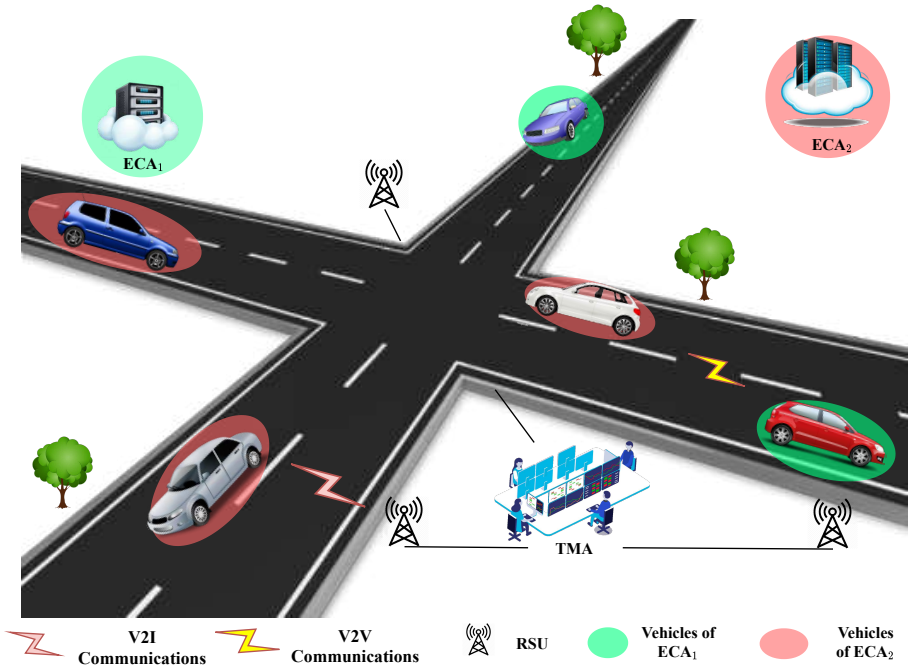

Figure 6.1: VANET Architecture

The role of each entity is detailed as follows:

1. Distributed KGCs: together, they are responsible for the registration of the ECAs. Moreover, in case of any misbehavior (e.g., false traffic-related messages), the KGCs collaborate to reveal the domain of the sending vehicle.

2. ECA: responsible for managing identities of the registered vehicles within the domain. In practice, it could act as the certificate authority managing certificates in the public key infrastructure system. In our application, the ECA is owned and controlled by vehicle enterprises that prefer to establish their own CCA and provide certificates to their vehicles [155].

3. RSU: a wireless communication device located on the roadside. It is securely connected to the TMA and serves as an intermediate device between the vehicle and the TMA. It utilizes the DSRC protocol for communication with vehicles.

4. OBU: embedded in the vehicle for transmitting traffic-related messages to other vehicles in its domain through V2V communications or to other RSUs through V2I communications. In case of traffic accidents, the OBU forwards traffic-related messages (e.g., car accidents and fires) to the TMA to prompt necessary actions within the required time.

6.2.2 Threat Model

In our proposed protocol, adversaries can be classified as internal or external. An internal adversary refers to entities, such as vehicles and RSUs (i.e., compromised RSU may want to impersonate vehicles or flood the network with false data), directly involved in VANETs. An external adversary refers to entities not directly involved in VANETs. Both internal and external adversaries can launch passive and active attacks. In passive attacks, adversaries monitor the V2V or V2I communication channel and gain access to confidential information, such as vehicle locations and plain texts of the sent messages. In active attacks, adversaries access the open wireless V2V or V2I communication channel to replay, modify, and fabricate transmitted messages over the channel, as in the Dolev-Yao threat model [44].

6.2.3 Security Requirements

Our Proposed protocol satisfies the following security requirements to ensure secure VANET communication.

1. Message authentication: the legitimacy of the sender of the message and the integrity of the sent data should be maintained.

2. Unlinkability: an adversary should not be able to link multiple requests sent by the same vehicle in order not to trace or profile any vehicle driver and launch a physical attack.

3. Resistance to replay attacks: an adversary cannot resend a valid message after a time interval. The receiver should reject the message.

4. Resistance to impersonation attack: upon intercepting a message sent with a pseudo-identity. An adversary cannot use this pseudo-identity again to impersonate a legal vehicle in future transactions.

5. Conditional privacy-preserving: upon a misconduct only distributed KGCs collaboratively can trace back the identity of the sender vehicle.

6.3 Protocol Design

Our protocol makes use of Schnorr Zero-Knowledge Proof of discrete logarithm knowledge [121] and Schnorr signature [120]. The Schnorr signature is used to authenticate the sender's vehicle. Each of the distributed KGCs generates a different signing key for the ECA during the registration phase. Then, the ECA computes its signing keys, collectively, from the issued signing keys. Later on, vehicles generate an instantaneous pseudo-identity that is valid for a certain period and can be used several times during its validity period. Then, the vehicle obtains an authentication token from their ECAs over

the instantaneous pseudo-identity of the vehicle. The vehicle uses such obtained authentication tokens to send its anonymous authenticated message. To maintain the anonymity and the unlinkability of the transmitted messages, the protocol randomizes the authentication message of the vehicle by multiplying it with a random number. Furthermore, the Schnorr ZKP is used to prove the knowledge of the random number that relates the randomized authentication token to our public system parameters without revealing it. Our protocol consists of a setup phase, ECA registration phase, vehicle signature generation phase, and message-sending phase. Moreover, CP-MAVE provides traceability of the ECA domain and traceability of the sender vehicle. Details of each phase are listed as follows.

6.3.1 Setup Phase

In this phase, distributed KGCs agree on the system public parameters, i.e., the used hash functions, the utilized elliptic curve, and the system public key. KGCs select the system parameters as follows. KGCs choose a 256-bit prime number p and an elliptic curve E over the finite field $\mathbb{F}_p$. KGCs choose a point $P \in E$ of order q as the generator point. It computes the system public $PK_g = PK_1 + PK_2 + \cdots + PK_{n_{GC}}$ where n_{GC} is the number of distributed KGCs and s_j and $PK_j = s_j P$ are the private and the public key of the j^{th} KGC, respectively. Moreover, KGCs agree on the collision-resistant one-way hash functions $H_1 : Z_q \rightarrow Z_q$, $H_2 : \{0,1\}^{128} \times \{0,1\}^{128} \times Z_q \times \mathcal{G} \rightarrow Z_q$, $H_3 : \{0,1\}^{128} \times \mathcal{G} \times \{0,1\}^{128} \times \mathcal{G} \rightarrow Z_q$, and $H_4 : \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \{0,1\} \times \{0,1\} \rightarrow Z_q$. Finally the public parameters of the system $params = (E,\ p,\ q,\ P,\ H_1,\ H_2,\ H_3,\ H_4,\ PK_g)$ are published.

6.3.2 ECA Registration with distributed KGCs

We assume that the registration phase runs in a secure environment. Distributed KGCs collaborate to issue the ECA signing key. Figure 6.2 illustrates the registration phase between an ECA_i. The ECA_i randomly selects $x_i \xleftarrow{r} Z_q$ as its partial private

key and computes the partial public key $X_i = x_i P$. Then, the ECA sends its identity ID_i along with the partial public key X_i to KGC_j. KGCs share the ID_i and X_i among themselves. After that, each KGC randomly selects $h_i^j, y_i^j \xleftarrow{r} Z_q$ and computes the ECA partial public key $Y_i^j = y_i^j P$ (i.e., the two partial keys X_i, Y_i are generated by the ECA and the distributed KGCs, respectively, and serve as the public key of the ECA). Then, KGC computes $y_i = \sum_j y_i^j$ and $C_i^j = h_i^j \sum_j Y_i^j$. Then, KGC_j issues the ECA signing key $\sigma_i^j = s_j + h_i^j \sum_j y_i^j + y_i^j$ which is verified on the ECA by validating $\sigma_i^j P \stackrel{?}{=} PK_j + C_i^j + Y_i^j$. Note that, KGC_j includes the secret s_j to indicate the issuing of the signing key by the j^{th} KGC. Then, the KGC stores $ID_i,\ X_i,\ Y_i^j,\ y_i^j,\ \sum_j y_i^j,\ h_i^j$. Note that, ECA_i obtains $\sigma_i^j,\ C_i^j,\ Y_i^j$ from other KGCs where $j = 1, 2, 3, \cdots, n_{GC}$. After that, ECA_i computes $\sigma_i = \sum_{j=1}^{n_{GC}} \sigma_i^j$, $C_i = \sum_{j=1}^{n_{GC}} C_i^j$, and $Y_i = \sum_{j=1}^{n_{GC}} Y_i^j$. Then, the ECA_i stores $ID_i,\ x_i,\ X_i,\ \sigma_i,\ C_i,\ Y_i$.

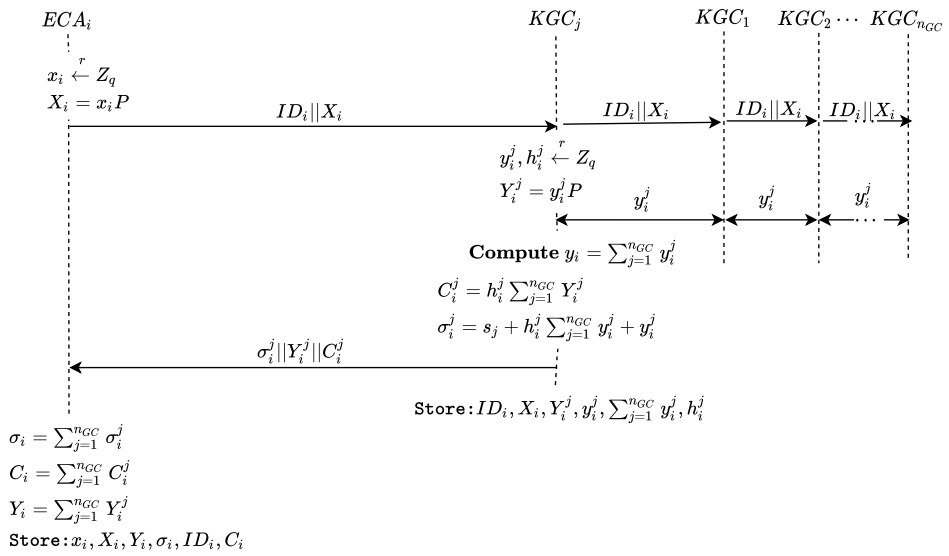

Figure 6.2: ECA registration

6.3.3 Vehicle's Signature Generation Phase

In the offline phase, a vehicle k obtains an authentication token from its ECA within the expiry time as shown in Fig. 6.3. It sends a Hello message to its ECA with

the receiver's identity (i.e., the TMA of the current city). The ECA replies with a nonce and the receiver's public key. Then, the sender computes $A = r_1 P$, where $r_1 \xleftarrow{r} Z_q$ is a random generated by the vehicle. Afterward, the ECA authenticates the sender and computes $h_{tr},\ B,\ I_p,\ \sigma_t$ where I_p is an integrity term on $ID_r||StartTime||ExpiryTime$ and h_{tr} is used for randomizing the public key of the ECA, X_i. Later, the sender computes I_p and verifies the issued authentication $\sigma_t P \stackrel{?}{=} PK_g + C_i + Y_i + I_p B + I_p X_i$. Note that, the $ExpiryTime$ is included to indicate the validity period of the issued authentication in case of multiple messages from the sender vehicle.

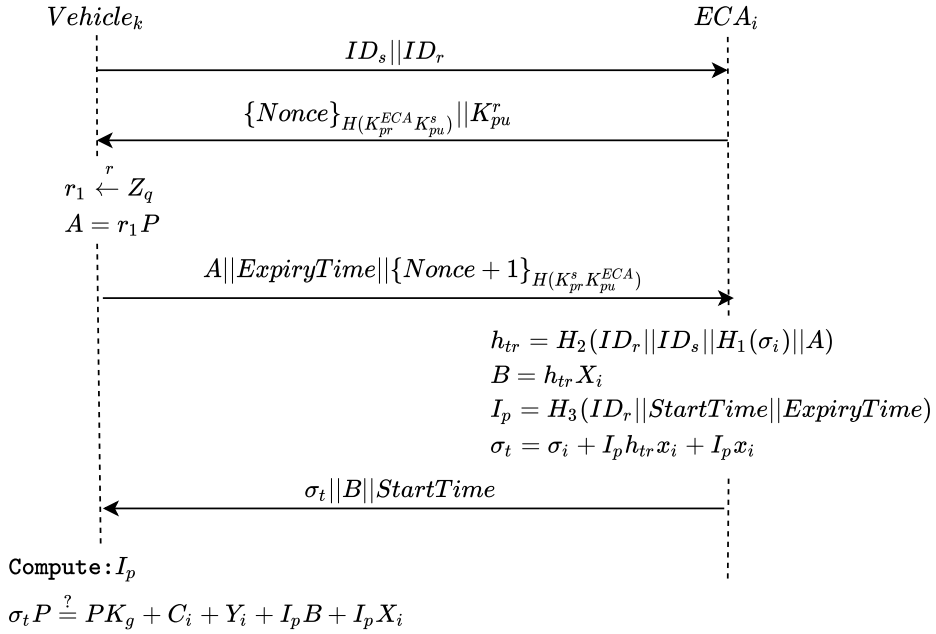

Figure 6.3: Vehicle obtaining signature from ECA

6.3.4 Message Sending Phase

In the online phase, for sending a traffic-related message to the city TMS, the sender vehicle generates $r_2,\ r_3 \xleftarrow{r} Z_q$, as shown in Fig. 6.4, and computes the random points $P_1 = r_2 PK_g,\ P_2 = r_2 C,\ P_3 = r_2 Y_i,\ P_4 = r_2 X_i,\ P_5 = r_2 B$, the commitment $T_1 = r_3 PK_g$ and the ZKP response $s_1 = r_3 + I_c r_2$ where $I_c = H_4(A||P_1||P_2||P_3||P_4||P_5||T_1||\{M\}_{H(r_1 K_{pu}^{r})}||$

$TimeStamp$). Then, it sends the transaction $\texttt{ID}_r||A||P_1||P_2||P_3||P_4||P_5||T_1||\sigma_2||\{M\}_{H(r_1K^r_{pu})}||$ $s_1||ID_r||StartTime||ExpiryTime$ to the receiver RSU which forwards it to TMA where $\{M\}_{H(r_1K^r_{pu})}$ is the traffic message encrypted using the symmetric key $r_1K^r_{pu}$, σ_2 is the signature over the message, $TimeStamp$ is the time stamp of the message to indicate the freshness of the message M and $A,\ P_1,\ P_2,\ P_3,\ P_4,\ P_5,\ T_1$ are points for verifying the signature σ_2. On the receiver side, the receiver computes I_c, I_p and verifies $s_1PK_g \stackrel{?}{=} T_1 + I_cP_1$ and $\sigma_2P \stackrel{?}{=} P_1 + P_2 + P_3 + I_pP_4 + I_pP_5 + I_cA$. Then, it retrieves the message M.

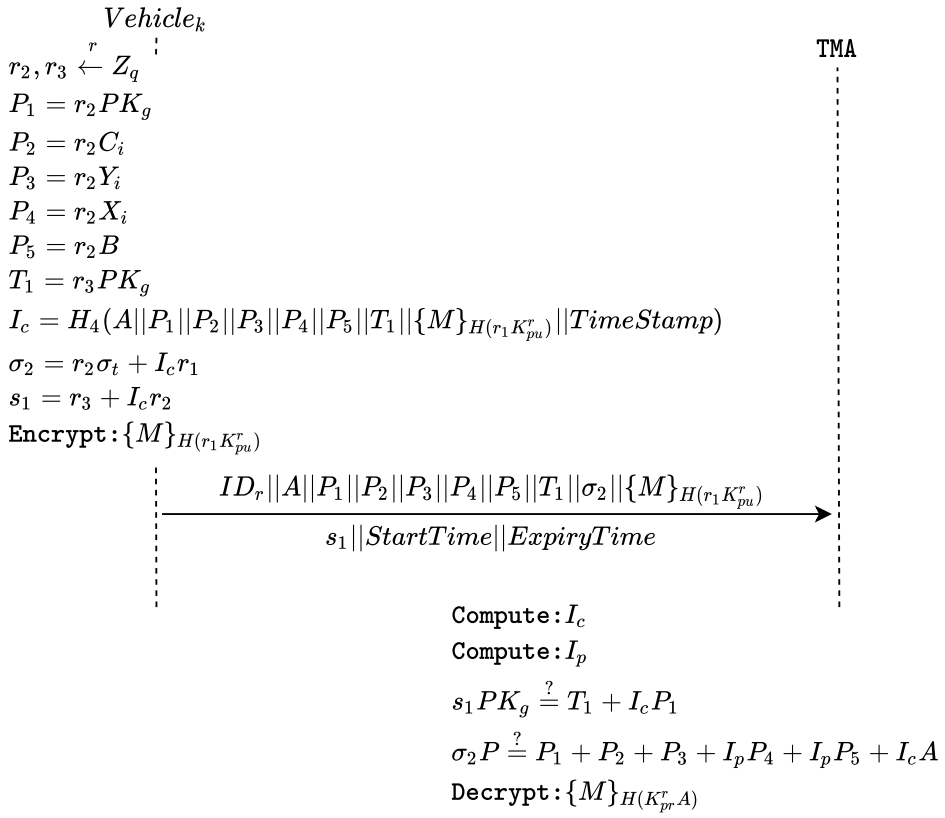

Figure 6.4: Sending a message from the vehicle to the TMA

6.3.5 Traceability of the Sender Vehicle ECA Domain

In case of a misbehaving vehicle (i.e., sending a false traffic message about a vehicle accident), the distributed KGCs, collaboratively, do an exhaustive search of complexity $\mathcal{O}$(n) to know the domain of the sender vehicle as follows. $P_2 = h_iP_3$ where

$i = 1, 2, \cdots, n_{ECA}$ is the index of the ECA and n_{ECA} is the total number of the registered ECAs within our protocol. Note that $h_i P_3 = \sum_{j=1}^{n_{GC}} h_i^j P_3$. After determining the ECA domain of the sender vehicle, the ECA does an exhaustive search over the registered vehicle to determine the sender vehicle's identity as follows. $P_5 = h_{tr}^k P_4$ where $h_{tr}^k = H_2(ID_r||ID_k||H_1(\sigma_i)||A)$, k is the index of the vehicle in the ECA's domain and $k = 1, 2, \cdots, n_{vehicle}$ while $n_{vehicle}$ is the total number of the registered vehicles. The identity of the claimed sender is checked $P_5 \stackrel{?}{=} H_2(ID_r||ID_k||H_1(\sigma_1)||A)P_4$. In case of not matching with any registered vehicle, the pre-determined ECA_i should be penalized since it generated a non-well-structured (i.e., $B \neq h_{tr}X_i$ or $\sigma_t \neq s_g + h_i y_i + y_i + I_p x_i + I_p h_{tr} x_i$) authentication token that used by the sender vehicle and passes the verification at the TMA side.

6.4 Security Analysis

In this section, we analyze the security properties of our proposed protocol. We formally prove the existential unforgeability of CP-MAVE against chosen message attacks based on the ECDLP assumption in the random oracle model. Moreover, we show the resistance of the proposed protocol to message modification attacks and replay attacks, and we demonstrate the conditional privacy-preserving of our proposed protocol. We proceed by modeling the adversarial capabilities then we define the existential unforgeability property of our protocol.

6.4.1 Security Definitions

We define the security model of CP-MAVE through a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ where they interact with the following queries.

- **Setup:** $\mathcal{C}$ generates the private key $s_g = s_1 + s_2 + s_3 + \cdots + s_{n_{GC}}$ and the system parameters $params = (E,\ p,\ q,\ P,\ H_1,\ H_2,\ H_3,\ H_4,\ PK_g)$. Then, $\mathcal{C}$ sends *Params*

to $\mathcal{A}$.

- **Sign:** Upon receiving $\mathcal{A}$'s query with the message M, $\mathcal{C}$ generates and returns $\mathtt{ID}_r||A||P_1||P_2||P_3||P_4||P_5||T_1||\sigma_2||\{M\}_{H(r_1 K_{pu}^r)}||s_1||ID_r||StartTime||ExpiryTime$ to $\mathcal{A}$.

- **Verify:** Upon receiving $\mathcal{A}$'s query with $\mathtt{ID}_r||A||P_1||P_2||P_3||P_4||P_5||T_1||\sigma_2||\{M\}_{H(r_1 K_{pu}^r)}$ $||s_1||ID_r||StartTime||ExpiryTime$, $\mathcal{C}$ returns TRUE to $\mathcal{A}$ if $s_1 PK_g \stackrel{?}{=} T_1 + I_c P_1$ and $\sigma_2 P \stackrel{?}{=} P_1 + P_2 + P_3 + I_p P_4 + I_p P_5 + I_c A$ to $\mathcal{A}$. Otherwise, it returns FALSE

In what follows, we define the existential unforgeability game for our proposed protocol.

Definition 6.1 *(Existential Unforgeability Against Chosen Message Attack): the challenger $\mathcal{C}$ runs the **Setup** to produce the private key and the system parameters params. The signing queries **Sign** are allowed for $\mathcal{A}$, i.e., $\mathcal{A}$ can make n_q signing queries on adaptively chosen message M_w where $w = 1, 2, \cdots, n_q$. For the signing query on M_w, $\mathcal{C}$ runs the signing procedure and sends transaction $\mathcal{S}_w = \mathtt{ID}_r||A||P_1||P_2||P_3||P_4||P_5||T_1||\sigma_w||\{M_w\}_{H(r_1 K_{pu}^r)}$ $||s_1||ID_r||StartTime||ExpiryTime$ to $\mathcal{A}$. At last, $\mathcal{A}$ is required to return a forged signature σ_y on M_y. $\mathcal{A}$ is considered to win the game if two conditions both satisfy, i.e., M_y has not been queried in previous signing queries and σ_y is a valid signature on M_y.*

Definition 6.2 *(Message Integrity MI): A PPT adversary $\mathcal{A}$, which plays the game defined in Definition 6.1, breaks the proposed CP-MAVE protocol if $\mathcal{A}$ can forge a valid signature on M_y with a success probability greater than ϵ. The proposed protocol is MI-secure if any adversary $\mathcal{A}$ cannot break the proposed protocol with a probability greater than ϵ.*

6.4.2 Formal Security Analysis

Using the security model discussed before, in what follows, we formally prove that our proposed protocol achieves existential unforgeability against the chosen message attacks in the random oracle model. Moreover, we show the resistance of the protocol to message modification and replay attacks. Furthermore, we demonstrate the privacy-preserving property and the traceability of the protocol.

Theorem 6.1 *Under the assumption of the intractability of ECDLP, the signature protocol CP-MAVE is MI-secure.*

Proof:

Suppose that a PPT adversary $\mathcal{A}$ can forge a message $\mathtt{ID}_r||A||P_1||P_2||P_3||P_4||P_5||T_1||\sigma_2||\{M\}_{H(r_1 K_{pu}^r)}||s_1||ID_r||StartTime||ExpiryTime$. We can construct an adversary $\mathcal{B}$ which can solve ECDLP with a negligible probability by running $\mathcal{A}$ as a subroutine. Given the EC point, $Q = sP$ such that $s \overset{r}{\leftarrow} Z_q$, $\mathcal{B}$ simulates the protocol and solves the ECDLP as follows.

- **Setup:** $\mathcal{B}$ sets the public key $PK_g = Q$ and sends $params = (E,\ p,\ q,\ P,\ H_1,\ H_2,\ H_3,\ H_4,\ PK_g)$ to $\mathcal{A}$.

- H_1**:** $\mathcal{B}$ maintains a list $\mathcal{L}_1$ with the form of (σ_1, k_1), which is initialized to be empty. Upon receiving $\mathcal{A}$'s query with σ_1, $\mathcal{B}$ checks the entry (σ_1) in $\mathcal{L}_1$ and returns $k_1 = H_1(\sigma_1)$ to $\mathcal{A}$. If the entry (σ_1) does not exist, $\mathcal{B}$ generates randomly $k_1 \leftarrow Z_q$ and updates $\mathcal{L}_1$ with $(\sigma_1), k_1$.

- H_2**:** $\mathcal{B}$ maintains a list $\mathcal{L}_2$ with the form of $(ID_r||ID_s||H_1(\sigma_1)||A, k_2)$, which is initialized to be empty. Upon receiving $\mathcal{A}$'s query with $ID_r||ID_s||H_1(\sigma_1)||A$, $\mathcal{B}$ checks the entry $(ID_r||ID_s||H_1(\sigma_1)||A)$ in $\mathcal{L}_2$ and returns $k_2 = H_2(ID_r||ID_s||H_1(\sigma_1)||A)$ to $\mathcal{A}$. If the entry $(ID_r||ID_s||H_1(\sigma_1)||A)$ does not exist, $\mathcal{B}$ generates randomly $k_2 \leftarrow Z_q$ and updates $\mathcal{L}_2$ with $(ID_r||ID_s||H_1(\sigma_1)||A, k_2$.

- H_3**:** $\mathcal{B}$ maintains a list $\mathcal{L}_3$ with the form of $(ID_r||A||Nonce||ExpiryTime||B)$, which is initialized to be empty. Upon receiving $\mathcal{A}$'s query with $ID_r||StartTime||ExpiryTime$, $\mathcal{B}$ checks the entry $(ID_r||StartTime||ExpiryTime)$ in $\mathcal{L}_3$ and returns $k_3 = H_3(ID_r||StartTime||ExpiryTime)$ to $\mathcal{A}$. If the entry $ID_r||StartTime||ExpiryTime$ does not exist, $\mathcal{B}$ generates randomly $k_3 \leftarrow Z_q$ and updates $\mathcal{L}_3$ with $(ID_r||StartTime||ExpiryTime), k_3$.

- H_4**:** $\mathcal{B}$ maintains a list $\mathcal{L}_4$ with the form of $(A||P_1||P_2||P_3||P_4||P_5||T_1||M_{H(r_1K_{pu}^r)})$, which is initialized to be empty. Upon receiving $\mathcal{A}$'s query with $A||P_1||P_2||P_3||P_4||P_5||T_1||M_{H(r_1K_{pu}^r)}$, $\mathcal{B}$ checks the entry $(A||P_1||P_2||P_3||P_4||P_5||T_1||M_{H(r_1K_{pu}^r)}$ in $\mathcal{L}_4$ and returns $r_4 = H_4(A||P_1||P_2||P_3||P_4||P_5||T_1||M_{H(r_1K_{pu}^r)})$ to $\mathcal{A}$. If the entry $A||P_1||P_2||P_3||P_4||P_5||T_1||M_{H(r_1K_{pu}^r)}$ does not exist, $\mathcal{B}$ generates randomly $k_4 \leftarrow Z_q$ and updates $\mathcal{L}_4$ with $(A||P_1||P_2||P_3||P_4||P_5||\ T_1||M_{H(r_1K_{pu}^r)}, k_4$.

- **Sign:** Upon receiving $\mathcal{A}$'s query with the message M, $\mathcal{B}$ selects random numbers $\sigma_1, \sigma_2, r_2, r_3, ID_k, h_{tr}, I_p, I_c, y_i$. Then, $\mathcal{B}$ chooses random points Y_1, X_1 and computes $P_1 = r_2Q$, $P_2 = r_2C$, $P_3 = r_2Y_1$, $P_4 = r_2X_1$, $P_5 = r_2h_{tr}X_1$, $T_1 = r_3Q$. $\mathcal{B}$ computes $A = I_c^{-1}(\sigma_2P - P_1 - P_2 - P_3 - I_ph_{tr}X_1 - I_pP_4)$. Then, $\mathcal{B}$ adds $ID_r||StartTime||ExpiryTime$ into $\mathcal{L}_3$, $(A||P_1||P_2||P_3||P_4||T_1||M_{H(r_1K_{pu}^r)}, r_4)$ into $\mathcal{L}_4$, and $(ID_r||ID_s||H_1(\sigma_1)||A, r_2)$ into $\mathcal{L}_2$. Finally, $\mathcal{B}$ sends $s_1 = r_3 + I_cr_2$ and $\mathtt{ID}_r||A||P_1||P_2||P_3||P_4||P_5||T_1||\sigma_2||\{M\}_{H(r_1K_{pu}^r)}||s_1||StartTime||ExpiryTime$ to $\mathcal{A}$.

After that, $\mathcal{A}$ sends $s_1 = r_3 + I_cr_2$ and $\mathtt{ID}_r||A||P_1||P_2||P_3||P_4||P_5||T_1||\sigma_2||\{M\}_{H(r_1K_{pu}^r)}$ $||s_1||StartTime||ExpiryTime$ to $\mathcal{B}$ which verifies that

$$\begin{aligned} \sigma_2P &= P_1 + P_2 + P_3 + I_pP_4 + I_pP_5 + I_cA \\ s_1 &= r_3 + I_cr_2 \end{aligned} \tag{6.1}$$

$\mathcal{B}$ discards the process in case of not matching. Note that, the signature generated by $\mathcal{A}$ is

indistinguishable from those generated by authenticated vehicles where $s_1 PK_g = T_1 + I_c P_1$ and $\sigma_2 P = P_1 + P_2 + P_3 + I_p P_4 + I_p P_5 + I_c A$

According to the forking lemma [114], $\mathcal{A}$ can output another valid message $s_1^* = r_3 + I_c^* r_2$ and $\texttt{ID}_r||A||P_1||P_2||P_3||P_4||P_5||T_1||\sigma_2^*||\{M\}_{H(r_1 K_{pu}^r)}||s_1$ $StartTime||ExpiryTime$ where $\sigma_2 \neq \sigma_2^*$ (i.e., using the same randomness) and

$$\begin{aligned} \sigma_2^* P &= P_1 + P_2 + P_3 + I_p^* P_4 + I_p^* P_5 + I_c^* A \\ s_1^* &= r_3 + I_c^* r_2 \end{aligned} \tag{6.2}$$

from (1) and (2),

$$\begin{aligned} r_3 &= (I_c - {I_c}^*)^{-1}(s_1 - s_1^*) \\ r_2(I_p - {I_p}^*)s_g &= (\sigma_2 - {\sigma_2}^*) - r_2(I_p - {I_p}^*)y_i(h_a + 1) \end{aligned} \tag{6.3}$$

At last $\mathcal{B}$ outputs $s_g = {r_2}^{-1}(I_p - {I_p}^*)^{-1}((\sigma_2 - {\sigma_2}') - r_2(I_p - I_p^*)y_i(h_a + 1))$ as a solution for the ECDLP for the given point Q. Nevertheless, under the assumption of the intractability of the ECDLP, $\mathcal{B}$ does not exist and accordingly our system achieves message integrity and is secure against chosen message attacks. ■

6.4.3 Unlinkability

The tuples $\texttt{ID}_r||A||P_1||P_2||P_3||P_4||P_5||T_1||\sigma_2||\{M\}_{H(r_1 K_{pu}^r)}||s_1||ID_r||StartTime$ $||ExpiryTime$ are sent to the TMA within each message transaction. Unlinkability between two messages transaction is maintained if an attacker cannot sufficiently distinguish if these two transactions are related or not [112]. The points $P_1,\ P_2,\ P_3,\ P_4,\ P_5$ are randomized with r_2 and T_1 is randomized with r_3. Note that, $r_2,\ r_3 \leftarrow Z_q$ are generated randomly with each authentication request. Moreover, the vehicle instantaneous pseudo-identity $A = r_1 P$ is generated randomly by the sender vehicle where $r_1 \leftarrow Z_q$. The sending vehicle sets its expiry time for such generated instantaneous pseudo-identity. Therefore, TMA and an attacker cannot link any messages sent from the same vehicle within different

expiry periods.

6.4.4 Conditional Privacy Preservation

Conditional privacy-preserving is also an important security aspect in our protocol where the sender's identity can be traced back in case of misbehavior (e.g., false traffic information, false location information). In CP-MAVE, distributed KGCs collaborate to get the ECA domain of the misbehaved vehicle by doing an exhaustive search as follows. $P_2 = h_i P_3$ where $h_i P_3 = \sum_{j=1}^{n_{GC}} h_i^j P_3$. After determining the ECA domain of the misbehaved vehicle, the ECA does an exhaustive search over registered vehicles with such ECA domain to determine the sender's identity by checking $h_{tr} \stackrel{?}{=} H_3(ID_r||ID_k||H_1(\sigma_1)||A)$. Therefore, the traceability of the real identity of malicious vehicles is guaranteed and CP-MAVE is deemed to be a conditional privacy-preserving protocol.

6.4.5 Resistance to Replay and Modification Attacks

This property is maintained in our protocol if the sent message M is not modified during its transmission to the TMA. [25]. Upon receiving an encryption of the message M, $\{M\}_{H(r_1 K_{pu}^r)}$, the TMA verifies $\sigma_2 P \stackrel{?}{=} P_1 + P_2 + P_3 + I_p P_4 + I_p P_5 + I_c A$ where $I_c = H_4(A||P_1||P_2||P_3||P_4||P_5||T_1||\{M\}_{H(r_1 K_{pu}^r)})$ and A is the instantaneous pseudo-identity of the sender vehicle. Any modification attack results in a new message M' with a new $I_c^* \neq I_c$ that does not pass the verification process $\sigma_2 P \stackrel{?}{=} P_1 + P_2 + P_3 + I_p P_4 + I_p P_5 + I_c A$. Also, such points A, B are instantaneous pseudo-identities generated randomly by the sender vehicle and the ECA, respectively. Note that A and B change with each message request. Therefore, any further replay of A and σ_t by any malicious adversary will not pass the verification of σ_2 with the new B' generated by the ECA. Moreover, the expiry time is included in the integrity term to prevent any reuse of σ_t out of the pre-determined expiry time by the ECA. Moreover, $TimeStamp$ is included to indicate the freshness of the message sent by the vehicle to the TMA.

6.5 Performance Evaluation

In what follows, we analyze the performance of our proposed protocol in terms of i) communication overhead (i.e., overhead messages exchanged between the sender vehicle and the TMA, other than the data message M), and ii) computation cost which involves operations that are done by the sender vehicle and the TMA during the communication process. In our performance analysis, we assume 128-bit random values and 128-bit ID. Also, we assume a 256-bit elliptic curve which typically provides nearly a 128-bit security level [5]. To perform the hash functions $H_1,\ H_2,\ H_3,\ H_4$ which incur EC points in their domains. Moreover, as the codomains for the hash function are in Z_q, we perform modular q operation on the output of the hash function where q is a 256-bit.

Table 6.2: Average cryptographic overhead time using 1000 runs

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

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

6.5.1 Computation Cost

The sender vehicle generates the random nonces $r_1,\ r_2,\ r_3$ and performs 7 scalar point multiplications to compute the randomized points $A,\ P_1,\ P_2,\ P_3,\ P_4,\ P_5$ and the commitment T_1. For verifying σ_t, the sender vehicle computes 1 hash operation and 3

Table 6.3: Performance comparison based on the computation complexity

Protocol	Required Cryptographic Operations	Total Time (msec)
[49]	4 T_e + 4 T_b	53.28
[144]	14 T_h+ 11 T_e	10.27
[145]	4 T_h+2 T_b + 2 T_e	26.84
[153]	4 T_h+ 5 T_{sm} + 7 T_b	89.2028
[140]	12 T_h+ 14 T_{sm} + 4 T_b + 2 T_e + 2 T_{pa}	57.6184
[50]	6 T_b+21 T_m + 20 T_e + 2 T_h	99.97
[17]	9 T_m+7 T_a + 6 T_h + 3 T_b	41.9992
[101]	9 T_m+4 T_h + 3 T_m + T_a + 3 T_b	42.1378
CP-MAVE	11 T_h + 4 T_{rng} + 20 T_{sm} + 6 T_s	9.4897

scalar point multiplications. For computing the signatures σ_2, the sender vehicle performs 1 hash operation. Moreover, for sending and receiving the data from the ECA and the TMA, the vehicle performs 2 symmetric Enc, 1 symmetric Dec, 2 hash, and 2 EC scalar multiplication to encrypt and decrypt the data.

On the other side, the TMA does 2 Hash operations and 6 EC scalar multiplications to verify the authentication of the message and 1 symmetric Dec operation. Moreover, it does 1 Hash and 1 EC scalar multiplication to retrieve the encrypted data.

The ECA performs 3 Hash operations to generate σ_t for the sender vehicle. Moreover, it performs 1 Hash and 1 EC scalar multiplication, 1 symmetric Enc, and 1 symmetric Dec, for sending its data encrypted to the vehicle.

6.5.2 Communication Overhead

The sender vehicle has to send the identity of the receiver ID_r of 128 bits, A, P_1, P_2, P_3, P_4, P_5, T_1 of 7×256 bits along with the σ_2 and s_1 of 2×128, the $StartTime$ and the $ExpiryTime$ of 2×128 bits which is equivalent to 304−byte communication overhead. Note that we use the EC point compression [133] for our representation of the EC point and we send only the $x-$ coordinate as a representation of the EC point. We also add an extra byte for determining the sign of the $y-$ coordinate for the 7 EC points.

6.5.3 Execution Time

We show the total computation-overhead time of the used cryptographic primitives in our protocol CP-MAVE compared to other related protocols in Fig. 6.5.

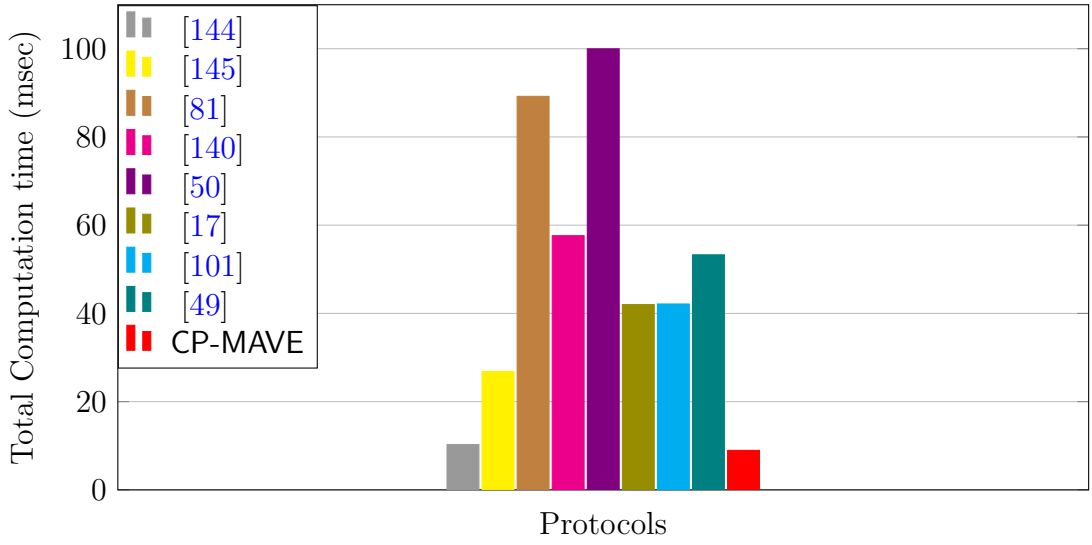

Figure 6.5: Evaluations of cryptographic primitives overhead time

In our comparison, we have considered both cross-domain protocols [49, 50] and blockchain-based protocols [17,49,81,140,144,145]. This is particularly relevant as blockchain protocols are known for their promising capabilities in facilitating cross-domain communication [33]. However, these capabilities often come with consensus overhead time, which may not align well with VANET applications [67]. On the other hand, CP-MAVE is blockchain-free, eliminating the need for consensus overhead. Moreover, it has the lowest computation time compared to other protocols and it provides the unlinkability of the sender vehicle to the TMA. In our comparison, we consider the number of the required cryptographic operations on the sender vehicle, ECA, and the TMA sides as reported in Table 6.3. We also report the average time required by each operation as shown in Table 6.2. Note that, we neglect modular operations (i.e., multiplication and addition) as they require microsecond execution time. For measuring the average execution time, we use a Raspberry Pi 4 Model B/8GB embedded with a 1.5 GHz 64-bit Quad-core ARM Cortex-A72 processor running the Raspbian 64-bit operating system. We run the cryptographic

primitives for 1000 times to compute the average execution time.

Moreover, to simulate the flow of messages among the sender vehicle, ECA, and the TMA, we implemented CP-MAVE using socket programming. The source code for the Python implementation of the socket programming experiment is available on the GitHub repository at https://github.com/M-Mahdy-Seif/CPMAVE. In our implementation, a Raspberry Pi 4 is used to simulate the OBU of the sender vehicle while two Lenovo ThinkStations with 16GB of memory and an Intel 8-Core i7-10700 CPU clocked at 2.90GHz acted as the ECA and the TMA, respectively. The experiment results in an average time of 43.21 milliseconds for the signature generation phase and 67.84 milliseconds for the message-sending phase. Thus, the total timing is about 111.05 milliseconds, which satisfies the DSRC requirements.

6.6 Summary

In this chapter, we introduced a vehicular communication protocol, named CP-MAVE, for secure cross-domain communication among vehicles and TMA in VANETs. CP-MAVE is an efficient conditional privacy-preserving protocol that achieves anonymity of the sender vehicle, message authentication, and integrity. CP-MAVE makes use of ECC for achieving an efficient 128-bit security level. To realize the traceability of misbehaved vehicles, CP-MAVE makes use of distributed KGCs to reach a consensus about the identity of the misbehaved vehicle. We have formally proved that under the hardness of the ECDLP, CP-MAVE achieves existential unforgeability against chosen message attacks. Furthermore, we evaluated CP-MAVE in terms of storage requirement, communication overhead, and computation cost requirements. Finally, we compared the execution time of our protocol with other close vehicular communication protocols.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The edge computing paradigm has been proposed to mitigate the single point of failure in the traditional IoT-cloud paradigm and to meet the expected QoS requirements for IoT applications. In such a paradigm, resource-constrained IoT devices outsource their heavy computation to a nearby edge node (e.g., routers, switches, and base stations) to avoid the high latency in the traditional IoT-cloud framework. While such a 3-tier architecture enables low-end IoT devices to outsource heavy computation to nearby edge nodes, it gives rise to several security challenges, especially for mobile IoT devices that are expected to be dynamically deployed anywhere.

A brief summary of the contributions accomplished in this thesis is as follows. In Chapter 3, we introduced SKAFS for secure computation offloading services between an IoT device and an IoT gateway in the IoT-edge-cloud paradigm. SKAFS achieves anonymity of the IoT device, forward secrecy, and resilience to hardware compromise and desynchronization-based DoS attacks. In Chapter 4, we presented SKICAP, an inter-cloud mutual authentication protocol. To mitigate impersonation and replay attacks, the temporary pseudo-identity of the IoT device is composed of chained authenticated

messages. In Chapter 5, we proposed MAPFS, a privacy-preserving mutual authentication protocol between the IoT device and the edge node for computation offloading services in the IoT-edge-cloud paradigm. MAPFS achieves anonymity of the IoT device, session unlinkability, and perfect forward secrecy for the established session key with revocation ability for the misbehaving IoT device. In Chapter 6, we introduced CP-MAVE, a vehicular communication protocol for secure cross-domain communication among vehicles and TMA in VANETs. CP-MAVE is an efficient conditional privacy-preserving protocol that achieves anonymity of the sender vehicle, message authentication, and integrity. To realize the traceability of misbehaved vehicles, CP-MAVE makes use of distributed KGCs to identify the misbehaved vehicle.

Moreover, we have provided performance analysis to show that the protocols are efficient in terms of storage, communication cost, and computation complexity, making them compatible with the IoT applications QoS. Furthermore, we implemented a prototype of our proposed protocols using socket programming to simulate the message flow between the protocol entities in a real-time experiment and calculate its end-to-end latency. Our source code for the Python implementation is available in the GitHub repository at https://github.com/M-Mahdy-Seif. Prototypes of the proposed protocols show that secure and lightweight mutual authentication protocols that consider the middle edge nodes and meet the required IoT QoS (i.e., latency) are achievable.

In this thesis, we addressed some of the rising challenges in adopting edge computing for IoT applications. More precisely, the vulnerability of the credentials of IoT devices in public open places is mitigated by embedding PUFs on the IoT device where credentials of the IoT device are not stored on the IoT device; instead, they are computed using PUF responses during the authentication process. Moreover, mutual authentication between IoT devices and edge nodes without the involvement of the CA in the authentication process can be addressed using Schnorr ZKP of discrete logarithm knowledge and Schnorr signatures. The mitigation of the single point of failure in the conventional IoT-cloud

and the realization of the distributed nature by allowing the IoT devices to obtain the CO service from nearby edge nodes is achieved in SKICAP and SKAFS. The realization of IoT-edge mutual authentication without the need for an online CA is achieved in MAPFS and CP-MAVE. From the presented protocols, we conclude that the realization of lightweight secure mutual authentication protocols among IoT and edge nodes for CO services, maintaining the confidentiality of the data and the anonymity of the IoT users, is feasible.

7.2 Future Work

In this section, we shed light on some directions for future research that can further extend the contributions of this thesis. In our future work, we will take into consideration the hardware compromise of edge nodes and the trustworthiness issue in the edge computations. We aim to

- mitigate the trustworthiness issue in the computation of the edge node by allowing a smart contract to verify edge computation and data storage through methods such as proof of data replication, proof of data possession, proofs of retrievability [129], and verifiable computation [66].

- protect the sensed data from unauthorized access by enforcing access control policies [47] over the outsourced sensed data. Such access control mechanisms, such as attribute-based access control, consist of different attributes that provide a flexible and fine-grained access policy.

- realize a distributed nature registration procedure without relying on a single CA for keeping the registration data of the IoT device. There is potential for using secure multi-party computation, yet the number of rounds and interactivity could be a challenging issue, especially in the traceability of misbehaving IoT devices.

Bibliography

[1] A bilinear pairing library for petlib. https://pypi.org/project/bplib/. [Online; accessed 13-August-2024].

[2] Benchmarking wolfssl and wolfcrypt. https://www.wolfssl.com/docs/benchmarks/. [Online; accessed 13-August-2024].

[3] Gartner says 5.8 billion enterprise and automotive IOT endpoints will be in use in 2020. https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io. [Online; accessed 26-November-2021].

[4] Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025(in billions). https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/. [Online; accessed 13-August-2024].

[5] Standards for Efficient Cryptography Group. https://www.secg.org/. [Online; accessed 13-August-2024].

[6] D. Abbasinezhad-Mood and M. Nikooghadam. An anonymous ECC-based self-certified key distribution scheme for the smart grid. *IEEE Transactions on Industrial Electronics*, 65(10):7996–8004, 2018.

[7] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *International conference on the theory and application of cryptology and information security*, pages 452–473. Springer, 2003.

[8] M. A. Al-Shareeda, M. Anbar, S. Manickam, and I. H. Hasbullah. A secure pseudonym-based conditional privacy-preservation authentication scheme in vehicular ad hoc networks. *Sensors*, 22(5):1696, 2022.

[9] I. Ali, Y. Chen, N. Ullah, R. Kumar, and W. He. An efficient and provably secure ECC-based conditional privacy-preserving authentication for vehicle-to-vehicle communication in VANETs. *IEEE Transactions on Vehicular Technology*, 70(2): 1278–1291, 2021.

[10] 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.

[11] 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, 2020.

[12] 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 *Computer Aided Verification: 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005. Proceedings 17*, pages 281–285. Springer, 2005.

[13] G. S. Aujla, R. Chaudhary, K. Kaur, S. Garg, N. Kumar, and R. Ranjan. SAFE: SDN-assisted framework for edge–cloud interplay in secure healthcare ecosystem. *IEEE Transactions on Industrial Informatics*, 15(1):469–480, 2018.

[14] G. Avoine, S. Canard, and L. Ferreira. Symmetric-key authenticated key exchange (SAKE) with perfect forward secrecy. In *Topics in Cryptology–CT-RSA 2020: The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24–28, 2020, Proceedings*, pages 199–224. Springer, 2020.

[15] M. Azees, P. Vijayakumar, and L. J. Deboarh. EAAP: Efficient anonymous authentication with conditional privacy-preserving scheme for vehicular ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2467–2476, 2017.

[16] M. Azees, P. Vijayakumar, M. Karuppiah, and A. Nayyar. An efficient anonymous authentication and confidentiality preservation schemes for secure communications in wireless body area networks. *Wireless Networks*, 27(3):2119–2130, 2021.

[17] P. Bagga, A. K. Sutrala, A. K. Das, and P. Vijayakumar. Blockchain-based batch authentication protocol for Internet of Vehicles. *Journal of Systems Architecture*, 113:101877, 2021.

[18] G. Bansal, N. Naren, V. Chamola, B. Sikdar, N. Kumar, and M. Guizani. Lightweight mutual authentication protocol for V2G using physical unclonable function. *IEEE Transactions on Vehicular Technology*, 69(7):7234–7246, 2020.

[19] Bellare. A note on negligible functions. *Journal of Cryptology*, 15:271–284, 2002.

[20] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni. A survey on fog computing for the internet of things. *Pervasive and mobile computing*, 52: 71–99, 2019.

[21] A. R. Biswas and R. Giaffreda. IoT and cloud convergence: Opportunities and challenges. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 375–376. IEEE, 2014.

[22] J. Bringer, H. Chabanne, and T. Icart. Improved privacy of the tree-based hash protocols using physically unclonable function. In *International Conference on Security and Cryptography for Networks*, pages 77–91. Springer, 2008.

[23] C. Brzuska, H. Jacobsen, and D. Stebila. Safely exporting keys from secure channels. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 670–698. Springer, 2016.

[24] 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.

[25] S. Čapkun, M. Čagalj, R. Rengaswamy, I. Tsigkogiannis, J.-P. Hubaux, and M. Srivastava. Integrity codes: Message integrity protection and authentication over insecure channels. *IEEE Transactions on Dependable and Secure Computing*, 5(4): 208–223, 2008.

[26] H. Chang, A. Hari, S. Mukherjee, and T. Lakshman. Bringing the cloud to the edge. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 346–351. IEEE, 2014.

[27] U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, R. S. Chakraborty, D. Mahata, and M. M. Prabhu. Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database. *IEEE transactions on dependable and secure computing*, 16(3):424–437, 2018.

[28] U. Chatterjee, R. Sadhukhan, V. Govindan, D. Mukhopadhyay, R. S. Chakraborty, S. Pati, D. Mahata, and M. M. Prabhu. PUFSSL: An OpenSSL extension for PUF based authentication. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pages 1–5. IEEE, 2018.

[29] R. Chaudhary, G. S. Aujla, S. Garg, N. Kumar, and J. J. Rodrigues. SDN-enabled multi-attribute-based secure communication for smart grid in IIoT environment. *IEEE Transactions on Industrial Informatics*, 14(6):2629–2640, 2018.

[30] D. Chaum and E. Van Heyst. Group signatures. In *Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings 10*, pages 257–265. Springer, 1991.

[31] C.-M. Chen and W.-C. Ku. Stolen-verifier attack on two new strong-password authentication protocols. *IEICE Transactions on communications*, 85(11):2519–2521, 2002.

[32] J. Chen, L. Wang, M. Wen, K. Zhang, and K. Chen. Efficient certificateless online/offline signcryption scheme for edge IoT devices. *IEEE Internet of Things Journal*, 9(11):8967–8979, 2021.

[33] J. Chen, Z. Zhan, K. He, R. Du, D. Wang, and F. Liu. Xauth: Efficient privacy-preserving cross-domain authentication. *IEEE Transactions on Dependable and Secure Computing*, 19(5):3301–3311, 2021.

[34] L. Chuat, A. Abdou, R. Sasse, C. Sprenger, D. Basin, and A. Perrig. SoK: Delegation and Revocation, the Missing Links in the Web's Chain of Trust. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 624–638. IEEE, 2020.

[35] M. Conti, A. Dehghantanha, K. Franke, and S. Watson. Internet of Things security and forensics: Challenges and opportunities. *Future Generation Computer Systems*, 78:544–546, 2018.

[36] C. Cremers and M. Feltz. Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In *Computer Security–ESORICS 2012: 17th*

European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings 17, pages 734–751. Springer, 2012.

[37] M. Dammak, S.-M. Senouci, M. A. Messous, M. H. Elhdhili, and C. Gransart. Decentralized lightweight group key management for dynamic access control in IoT environments. *IEEE Transactions on Network and Service Management*, 17(3): 1742–1757, 2020.

[38] M. Debe, K. Salah, M. H. U. Rehman, and D. Svetinovic. IoT public fog nodes reputation system: A decentralized solution using Ethereum blockchain. *IEEE Access*, 7:178082–178093, 2019.

[39] M. Debe, K. Salah, M. H. U. Rehman, and D. Svetinovic. Blockchain-Based Decentralized Reverse Bidding in Fog Computing. *IEEE Access*, 8:81686–81697, 2020.

[40] M. Debe, K. Salah, M. H. U. Rehman, and D. Svetinovic. Monetization of services provided by public fog nodes using blockchain and smart contracts. *IEEE Access*, 8:20118–20128, 2020.

[41] J. Delvaux. Machine-learning attacks on polypufs, ob-pufs, rpufs, lhs-pufs, and puf–fsms. *IEEE Transactions on Information Forensics and Security*, 14(8):2043–2058, 2019.

[42] W. Diffie and M. E. Hellman. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pages 365–390. 2022.

[43] 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.

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

[45] S. Dong, H. Yang, J. Yuan, L. Jiao, A. Yu, and J. Zhang. Blockchain-based cross-domain authentication strategy for trusted access to mobile devices in the IoT. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1610–1612. IEEE, 2020.

[46] R. G. Engoulou, M. Bellaïche, S. Pierre, and A. Quintero. VANET security surveys. *Computer Communications*, 44:1–13, 2014.

[47] V. Ezhil Arasi, K. Indra Gandhi, and K. Kulothungan. Auditable attribute-based data access control using blockchain in cloud storage. *The Journal of Supercomputing*, 78(8):10772–10798, 2022.

[48] Q. Fan, J. Chen, M. Shojafar, S. Kumari, and D. He. SAKE*: A symmetric authenticated key exchange protocol with perfect forward secrecy for industrial Internet of Things. *IEEE transactions on industrial informatics*, 18(9):6424–6434, 2022.

[49] C. Feng, B. Liu, Z. Guo, K. Yu, Z. Qin, and K.-K. R. Choo. Blockchain-based cross-domain authentication for intelligent 5G-enabled internet of drones. *IEEE Internet of Things Journal*, 9(8):6224–6238, 2021.

[50] X. Feng, Q. Shi, Q. Xie, and L. Wang. P2BA: A privacy-preserving protocol with batch authentication against semi-trusted RSUs in vehicular ad hoc networks. *IEEE Transactions on Information Forensics and Security*, 16:3888–3899, 2021.

[51] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[52] H. S. Galal, M. ElSheikh, and A. M. Youssef. An efficient micropayment channel on ethereum. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26–27, 2019, Proceedings 14*, pages 211–218. Springer, 2019.

[53] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.

[54] 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, 2002.

[55] B. L. P. Gassend. *Physical random functions*. PhD thesis, Massachusetts Institute of Technology, 2003.

[56] N. Gayathri, G. Thumbur, P. R. Kumar, M. Z. U. Rahman, P. V. Reddy, et al. Efficient and secure pairing-free certificateless aggregate signature scheme for healthcare wireless medical sensor networks. *IEEE Internet of Things Journal*, 6(5):9064–9075, 2019.

[57] M. Ghamari, B. Janko, R. S. Sherratt, W. Harwin, R. Piechockic, and C. Soltanpur. A survey on wireless body area networks for ehealthcare systems in residential environments. *Sensors*, 16(6):831, 2016.

[58] P. Gope. PMAKE: Privacy-aware multi-factor authenticated key establishment scheme for advance metering infrastructure in smart grid. *Computer Communications*, 152:338–344, 2020.

[59] P. Gope, Y. Gheraibia, S. Kabir, and B. Sikdar. A secure IoT-based modern healthcare system with fault-tolerant decision making process. *IEEE Journal of Biomedical and Health Informatics*, 25(3):862–873, 2020.

[60] P. Gope and T. Hwang. Lightweight and energy-efficient mutual authentication and key agreement scheme with user anonymity for secure communication in global mobility networks. *IEEE Systems Journal*, 10(4):1370–1379, 2015.

[61] 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.

[62] 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.

[63] P. Gope and B. Sikdar. An efficient privacy-preserving authenticated key agreement scheme for edge-assisted internet of drones. *IEEE Transactions on Vehicular Technology*, 69(11):13621–13630, 2020.

[64] P. Gope and B. Sikdar. A privacy-aware reconfigurable authenticated key exchange scheme for secure communication in smart grids. *IEEE Transactions on Smart Grid*, 12(6):5335–5348, 2021.

[65] P. Gope, B. Sikdar, and O. Millwood. A scalable protocol level approach to prevent machine learning attacks on PUF-based authentication mechanisms for Internet-of-Medical-Things. *IEEE Transactions on Industrial Informatics*, page 1, 2021.

[66] L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, M. Schofnegger, and R. Walch. Reinforced concrete: a fast hash function for verifiable computation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1323–1335, 2022.

[67] J. Grover. Security of Vehicular Ad Hoc Networks using blockchain: A comprehensive review. *Vehicular Communications*, 34:100458, 2022.

[68] S. Guilley and R. Pacalet. SoCs security: a war against side-channels. In *Annales des télécommunications*, volume 59, pages 998–1009. Springer, 2004.

[69] B. B. Gupta, A. Gaurav, C.-H. Hsu, and B. Jiao. Identity-based authentication mechanism for secure information sharing in the maritime transport system. *IEEE Transactions on Intelligent Transportation Systems*, 24(2):2422–2430, 2021.

[70] B. Hammi, A. Fayad, R. Khatoun, S. Zeadally, and Y. Begriche. A lightweight ECC-based authentication scheme for Internet of Things (IoT). *IEEE Systems Journal*, 14(3):3440–3450, 2020.

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

[72] M. Hansen and H. Tschofenig. Terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. *draft-hansen-privacy-terminology-02 (work in progress)*, 2011.

[73] X. Hao, W. Ren, Y. Fei, T. Zhu, and K.-K. R. Choo. A blockchain-based cross-domain and autonomous access control scheme for internet of things. *IEEE Transactions on Services Computing*, 16(2):773–786, 2022.

[74] H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti. VANet security challenges and solutions: A survey. *Vehicular Communications*, 7:7–20, 2017.

[75] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran. The role of edge computing in internet of things. *IEEE communications magazine*, 56(11):110–115, 2018.

[76] C. Hristea and F. L. Ţiplea. A PUF-based destructive private mutual authentication RFID protocol. In *International Conference on Security for Information Technology and Communications*, pages 331–343. Springer, 2018.

[77] M. H. Ibrahim. Octopus: An Edge-fog Mutual Authentication Scheme. *IJ Network Security*, 18(6):1089–1101, 2016.

[78] S. H. Islam. Provably secure dynamic identity-based three-factor password authentication scheme using extended chaotic maps. *Nonlinear Dynamics*, 78(3):2261–2276, 2014.

[79] S. Jegadeesan, M. Azees, N. R. Babu, U. Subramaniam, and J. D. Almakhles. EPAW: Efficient privacy preserving anonymous mutual authentication scheme for wireless body area networks (WBANs). *IEEE Access*, 8:48576–48586, 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, 2019.

[81] X. Jia, M. Luo, K.-K. R. Choo, L. Li, and D. He. A Redesigned Identity-Based Anonymous Authentication Scheme for Mobile-Edge Computing. *IEEE Internet of Things Journal*, 9(12):10108–10120, 2021.

[82] Y. Jiang, K. Zhang, Y. Qian, and L. Zhou. Anonymous and efficient authentication scheme for privacy-preserving distributed learning. *IEEE Transactions on Information Forensics and Security*, 17:2227–2240, 2022.

[83] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao. Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory. *IEEE Transactions on Vehicular Technology*, 68 (3):2906–2920, 2019.

[84] A. Karati, S. H. Islam, and M. Karuppiah. Provably secure and lightweight certificateless signature scheme for IIoT environments. *IEEE Transactions on Industrial Informatics*, 14(8):3701–3711, 2018.

[85] M. Kaveh and M. R. Mosavi. A lightweight mutual authentication for smart grid neighborhood area network communications based on physically unclonable function. *IEEE Systems Journal*, 14(3):4535–4544, 2020.

[86] M. S. Kirkpatrick, S. Kerr, and E. Bertino. System on chip and method for cryptography using a physically unclonable function, June 10 2014. US Patent 8,750,502.

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

[88] C.-C. Lee, M.-S. Hwang, and I.-E. Liao. Security enhancement on a new authentication scheme with anonymity for wireless environments. *IEEE Transactions on Industrial Electronics*, 53(5):1683–1687, 2006.

[89] 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.

[90] M. Li, D. Hu, C. Lal, M. Conti, and Z. Zhang. Blockchain-enabled secure energy trading with verifiable fairness in industrial internet of things. *IEEE Transactions on Industrial Informatics*, 16(10):6564–6574, 2020.

[91] M. Li, L. Zhu, and X. Lin. Coride: A privacy-preserving collaborative-ride hailing service using blockchain-assisted vehicular fog computing. In *International Conference on Security and Privacy in Communication Systems*, pages 408–422. Springer, 2019.

[92] S. Li, T. Zhang, B. Yu, and K. He. A provably secure and practical PUF-based end-to-end mutual authentication and key exchange protocol for IoT. *IEEE Sensors Journal*, 21(4):5487–5501, 2020.

[93] X. Li, L. Zhu, X. Chu, and H. Fu. Edge computing-enabled wireless sensor networks for multiple data collection tasks in smart agriculture. *Journal of Sensors*, 2020(1): 4398061, 2020.

[94] Y. Li, Q. Cheng, X. Liu, and X. Li. A secure anonymous identity-based scheme in new authentication architecture for mobile edge computing. *IEEE Systems Journal*, 15(1):935–946, 2020.

[95] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang. Consortium blockchain for secure energy trading in industrial internet of things. *IEEE transactions on industrial informatics*, 14(8):3690–3700, 2017.

[96] Z. Liao, X. Pang, J. Zhang, B. Xiong, and J. Wang. Blockchain on security and forensics management in edge computing for IoT: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 19(2):1159–1175, 2021.

[97] T. Limbasiya and D. Das. Vcom: Secure and efficient vehicle-to-vehicle message communication protocol. *IEEE Transactions on Network and Service Management*, 18(2):2365–2376, 2020.

[98] H. Liu, Y. Zhang, and T. Yang. Blockchain-enabled security in electric vehicles cloud and edge computing. *IEEE Network*, 32(3):78–83, 2018.

[99] R. Maes and I. Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. *Towards Hardware-Intrinsic Security: Foundations and Practice*, pages 3–37, 2010.

[100] S. Matsumoto and R. M. Reischuk. IKP: Turning a PKI around with decentralized automated incentives. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 410–426. IEEE, 2017.

[101] C. Maurya and V. K. Chaurasiya. Efficient anonymous batch authentication scheme with conditional privacy in the Internet of Vehicles (IoV) applications. *IEEE Transactions on Intelligent Transportation Systems*, 24(9):9670–9683, 2023.

[102] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[103] R. C. Merkle. Protocols for public key cryptosystems. In *Secure communications and asymmetric cryptosystems*, pages 73–104. Routledge, 2019.

[104] S. Mohamed, N. Mouna, Y. Amr, and A. Riham. A lightweight authentication and inter-cloud payment protocol for edge computing. In *2020 IEEE 9th International Conference on Cloud Networking (CLOUDNET)*, pages 63–67. IEEE, 2020.

[105] D. Molnar, A. Soppera, and D. Wagner. A scalable, delegatable pseudonym protocol enabling ownership transfer of rfid tags. In *Selected Areas in Cryptography: 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers 12*, pages 276–290. Springer, 2006.

[106] 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.

[107] 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.

[108] J. Ni, K. Zhang, X. Lin, and X. Shen. Securing fog computing for internet of things applications: Challenges and solutions. *IEEE Communications Surveys & Tutorials*, 20(1):601–628, 2017.

[109] 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.

[110] W. Othman, M. Fuyou, K. Xue, and A. Hawbani. Physically secure lightweight and privacy-preserving message authentication protocol for VANET in smart city. *IEEE Transactions on Vehicular Technology*, 70(12):12902–12917, 2021.

[111] J. Pan and J. McElhannon. Future edge cloud and edge computing for internet of things applications. *IEEE Internet of Things Journal*, 5(1):439–449, 2017.

[112] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity—a proposal for terminology. In *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability Berkeley, CA, USA, July 25–26, 2000 Proceedings*, pages 1–9. Springer, 2001.

[113] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *International conference on the theory and applications of cryptographic techniques*, pages 387–398. Springer, 1996.

[114] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of cryptology*, 13(3):361–396, 2000.

[115] S. Rajamanickam, S. Vollala, R. Amin, and N. Ramasubramanian. Insider attack protection: Lightweight password-based authentication techniques using ECC. *IEEE Systems Journal*, 14(2):1972–1983, 2019.

[116] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, pages 552–565. Springer, 2001.

[117] 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, 2010.

[118] B. Safaei, A. M. H. Monazzah, M. B. Bafroei, and A. Ejlali. Reliability side-effects in Internet of Things application layer protocols. In *2017 2nd International Conference on System Reliability and Safety (ICSRS)*, pages 207–212. IEEE, 2017.

[119] N. Saxena and B. J. Choi. Integrated distributed authentication protocol for smart grid communications. *IEEE Systems Journal*, 12(3):2545–2556, 2016.

[120] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*, pages 239–252. Springer, 1989.

[121] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[122] S. Schulz, A.-R. Sadeghi, and C. Wachsmann. Short paper: Lightweight remote attestation using physical functions. In *Proceedings of the fourth ACM conference on Wireless network security*, pages 109–114, 2011.

[123] 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.

[124] M. Seifelnasr, R. AlTawy, and A. Youssef. Skafs: Symmetric key authentication protocol with forward secrecy for edge computing. *IEEE Internet of Things Journal*, 11(1):510–525, 2023.

[125] M. Seifelnasr, R. AlTawy, A. Youssef, and E. Ghadafi. Privacy-Preserving Mutual Authentication Protocol With Forward Secrecy for IoT-Edge-Cloud. *IEEE Internet of Things Journal*, 11(5):8105–8117, 2024.

[126] M. Seifelnasr, H. S. Galal, and A. M. Youssef. Scalable open-vote network on ethereum. In *Financial Cryptography and Data Security: FC 2020 International Workshops, AsiaUSEC, CoDeFi, VOTING, and WTSC, Kota Kinabalu, Malaysia, February 14, 2020, Revised Selected Papers 24*, pages 436–450. Springer, 2020.

[127] 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.

[128] J. Sepulveda, F. Willgerodt, and M. Pehl. SEPUFSoC: Using PUFs for memory integrity and authentication in multi-processors system-on-chip. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, pages 39–44, 2018.

[129] J. Shen, X. Chen, X. Huang, and Y. Xiang. Public proofs of data replication and retrievability with user-friendly replication. *IEEE Transactions on Dependable and Secure Computing*, 21(4):2047–2067, 2023.

[130] J. Shen, Z. Gui, S. Ji, J. Shen, H. Tan, and Y. Tang. Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks. *Journal of Network and Computer Applications*, 106:117–123, 2018.

[131] M. Shen, H. Lu, F. Wang, H. Liu, and L. Zhu. Secure and efficient blockchain-assisted authentication for edge-integrated Internet-of-Vehicles. *IEEE Transactions on Vehicular Technology*, 71(11):12250–12263, 2022.

[132] S. Shihab and R. AlTawy. Lightweight authentication scheme for healthcare with robustness to desynchronization attacks. *IEEE Internet of Things Journal*, 10(20): 18140–18153, 2023.

[133] A. Silverberg. Compression for trace zero subgroups of elliptic curves. *Trends in Mathematics*, 8:93–100, 2005.

[134] J. H. Silverman, J. Pipher, and J. Hoffstein. *An introduction to mathematical cryptography*, volume 1. Springer, 2008.

[135] 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.

[136] P. N. Smart. *Cryptography made simple*. Springer, 2016.

[137] J. Srinivas, S. Mukhopadhyay, and D. Mishra. Secure and efficient user authentication scheme for multi-gateway wireless sensor networks. *Ad Hoc Networks*, 54: 147–169, 2017.

[138] S. Sutar, A. Raha, and V. Raghunathan. D-PUF: An intrinsically reconfigurable DRAM PUF for device authentication in embedded systems. In *2016 International Conference on Compliers, Architectures, and Sythesis of Embedded Systems (CASES)*, pages 1–10. IEEE, 2016.

[139] A. K. Sutrala, P. Bagga, A. K. Das, N. Kumar, J. J. Rodrigues, and P. Lorenz. On the design of conditional privacy preserving batch verification-based authentication scheme for internet of vehicles deployment. *IEEE Transactions on Vehicular Technology*, 69(5):5535–5548, 2020.

[140] H. Tan and I. Chung. Secure authentication and key management with blockchain in VANETs. *IEEE access*, 8:2482–2498, 2019.

[141] P. Tedeschi, S. Sciancalepore, A. Eliyan, and R. Di Pietro. LiKe: Lightweight certificateless key agreement for secure IoT communications. *IEEE Internet of Things Journal*, 7(1):621–638, 2019.

[142] J.-L. Tsai and N.-W. Lo. Secure anonymous key distribution scheme for smart grid. *IEEE transactions on smart grid*, 7(2):906–914, 2015.

[143] 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.

[144] C. Wang, R. Huang, J. Shen, J. Liu, P. Vijayakumar, and N. Kumar. A novel lightweight authentication protocol for emergency vehicle avoidance in VANETs. *IEEE Internet of Things Journal*, 8(18):14248–14257, 2021.

[145] C. Wang, J. Shen, J.-F. Lai, and J. Liu. B-TSCA: Blockchain assisted trustworthiness scalable computation for V2I authentication in VANETs. *IEEE Transactions on Emerging Topics in Computing*, 9(3):1386–1396, 2020.

[146] 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.

[147] P. Wang and Y. Liu. SEMA: Secure and efficient message authentication protocol for VANETs. *IEEE systems journal*, 15(1):846–855, 2021.

[148] M. Wazid, A. K. Das, S. Shetty, J. JPC Rodrigues, and Y. Park. LDAKM-EIoT: Lightweight device authentication and key management mechanism for edge-based IoT deployment. *Sensors*, 19(24):5539, 2019.

[149] L. Wei, J. Cui, Y. Xu, J. Cheng, and H. Zhong. Secure and lightweight conditional privacy-preserving authentication for securing traffic emergency messages in

VANETs. *IEEE Transactions on Information Forensics and Security*, 16:1681–1695, 2020.

[150] L. Wei, J. Cui, H. Zhong, I. Bolodurina, and L. Liu. A lightweight and conditional privacy-preserving authenticated key agreement scheme with multi-TA model for fog-based VANETs. *IEEE Transactions on Dependable and Secure Computing*, 20 (1):422–436, 2021.

[151] S. Wolfert, L. Ge, C. Verdouw, and M.-J. Bogaardt. Big data in smart farming–a review. *Agricultural systems*, 153:69–80, 2017.

[152] D. Yamamoto, K. Sakiyama, M. Iwamoto, K. Ohta, M. Takenaka, and K. Itoh. Variety enhancement of PUF responses using the locations of random outputting RS latches. *Journal of Cryptographic Engineering*, 3(4):197–211, 2013.

[153] 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*, 23(2):1284–1298, 2020.

[154] L. Yang and J. Liu. Cross Domain Authentication Based on Blockchain for Mobile Terminals in Edge Computing Environment. In *2021 16th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pages 525–529. IEEE, 2021.

[155] Y. Yang, L. Wei, J. Wu, C. Long, and B. Li. A blockchain-based multidomain authentication scheme for conditional privacy preserving in vehicular ad-hoc network. *IEEE Internet of Things Journal*, 9(11):8078–8090, 2021.

[156] 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, 2020.

[157] B. Ying and A. Nayak. Lightweight remote user authentication protocol for multi-server 5G networks using self-certified public key cryptography. *Journal of Network and Computer Applications*, 131:66–74, 2019.

[158] C. Zhang, L. Zhu, C. Xu, C. Zhang, K. Sharif, H. Wu, and H. Westermann. BSFP: Blockchain-Enabled Smart Parking with Fairness, Reliability and Privacy Protection. *IEEE Transactions on Vehicular Technology*, 69(6):6578–6591, 2020.

[159] Y. Zheng and C.-H. Chang. Secure mutual authentication and key-exchange protocol between PUF-embedded IoT endpoints. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2021.

[160] X. Zhou, M. Luo, P. Vijayakumar, C. Peng, and D. He. Efficient certificateless conditional privacy-preserving authentication for VANETs. *IEEE Transactions on Vehicular Technology*, 71(7):7863–7875, 2022.

[161] M. Zhu, X.-Y. Liu, F. Tang, M. Qiu, R. Shen, W. Shu, and M.-Y. Wu. Public vehicles for future urban transportation. *IEEE transactions on intelligent transportation systems*, 17(12):3344–3353, 2016.

[162] M. Zhu, X.-Y. Liu, and X. Wang. An online ride-sharing path-planning strategy for public vehicle systems. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):616–627, 2018.