MESSAGE SUBSET INTEGRITY PROBLEM IN THE INTERNET OF THINGS

HAOTIAN YAN

PhD

The Hong Kong Polytechnic University

2024

The Hong Kong Polytechnic University

Department of Electrical and Electronic Engineering

Message Subset Integrity Problem in the Internet of Things

Haotian Yan

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

May 2024

CERTIFICATE OF ORIGINALITY

Abstract

With the fast development of the Internet of Things (IoT) applications and technologies, many new wireless communication protocols and hardware have emerged. However, there is one essential security objective during the message transmission in IoT, namely, message integrity, where an adversary cannot modify the message. Classic cryptographic solutions to integrity include the message authentication code and digital signature. In the IoT scenario, especially fog and edge computing, however, different devices have different computation abilities: a fog or edge node may have a strong computation ability to handle many computation tasks, while a sensor node cannot. Therefore, a participant with limited computation ability cannot verify all incoming messages. To solve this problem, we can segment the message into different subsets so that the devices with varying computation abilities can select one or more of the message subsets for verification.

The works described in this thesis are mainly divided into three parts. First, we try to solve the subset privacy problem, which is a by-product of the subset integrity. In some cases, a subset of the message also has privacy concerns where the message may contain useless (in terms of integrity) but sensitive content. Unfortunately, existing MACs or signatures only address message integrity while ignoring the privacy problem. To address the problem, we provide a novel scheme so that only one MAC is necessary for protecting the message integrity, even if the privacy concerns from different clients are varied. Second, we provide a new scheme to support the subset integrity verification between two entities in the fog based industrial IoT scenario. Insides, the fog node has great computation ability. Thus, it can randomly and optimally verify a subset of the message and leave the remaining subsets verified by the receiver. Finally, we extend our research to a smart city scenario, where several edge nodes are involved in subset verification. Since a message is separated into different subsets, several edge nodes should cooperate to verify the message. Therefore, a novel signature scheme is necessary. Moreover, how this process can be optimized in terms of verification latency and system throughput is an open problem. The optimization is incredibly challenging when the verification order of various edge nodes is essential.

Publications Arising from the Thesis

1. Qingqing Ye, Haibo Hu, Ninghui Li, Xiaofeng Meng, Huadi Zheng, and Haotian Yan "Beyond value perturbation: Local differential privacy in the temporal setting", *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Virtual, May 2021.

2. Haotian Yan, Haibo Hu, Qingqing Ye, and Tang Li, "SPMAC: Scalable Prefix Verifiable Message Authentication Code for Internet of Things" *IEEE Transactions on Network and Service Management (TNSM)*, Volume: 19, Issue: 3, pp.3453-3464, September 2022.

3. Haotian Yan, Haibo Hu, and Qingqing Ye, "Partial message verification in fog-based industrial Internet of things" *Computers & Security*, Volume 135: 103530, December 2023.

4. Haotian Yan, Haibo Hu, and Qingqing Ye, "Time-Specific Integrity Service in MQTT Protocol" *The ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, (pp. 113-125). IEEE.

5. Haotian Yan, Haibo Hu, Qingqing Ye, and Jianliang Xu, "Multi-hop Sanitizable Signature for Collaborative Edge Computing" *Journal of Computer Security*, Preprint (2024): 1-27.

Acknowledgments

First of all, I should express my great gratitude to my supervisor, Prof. Haibo Hu, for supporting my research work from 2019. He always gives me constructive advice on my work and is patient in answering the questions I have been confronted with during these years. He offers me an essential opportunity to pursue a PhD degree in computer science. Without his help, I may still be confused about my research career in the future. Although he sometimes puts some pressure on my research, I know that he wants to make my research work better. His critical judgment and rigorous logic in the research area will inspire me in my future work.

Second, I am grateful to Dr. Qingqing Ye for having a discussion with me about the research work. Her experience in research impresses me and becomes a role model for me. Also, I want to thank the teammates in the ASTAPLE group. The senior members, Dr. Huadi Zheng, Dr. Ziyang Han, and Dr. Tang Li, kindly helped me when I encountered some troubles when starting the research. I will also appreciate for the company of members for Miss. Rong Du, Miss. Yaxin Xiao and Mr. Yue Fu. Without them, I would not have had a good learning environment in these years. Moreover, the ASTAPLE group has grown larger over these years. I should also thank for the assistance of them, Dr. Sen Zhang, Mr. Jiawei Duan, Miss. Li Bai, Mr. Haoyang Li, Miss. Yulian Mao, Mr. Ronghua Li, Mr. Xun Ran, Mr. Xinwei Zhang, and Mr. Chun Ho Kong for bringing joy to this office.

Last but not least, I should thank for the assistance from my parents. I have been studied in Hong Kong for six years. During the years, I live here without the accompany of the family, and confront so many things. They gave me great power so that I can get through that serious time.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the prosperity of communication technology and artificial intelligence, data has become increasingly valuable, and so has data transmission. Consequently, the Internet of Things (IoT) has become a hot topic. According to the research in IoT Analytics, the annual growth rate of the IoT market has reached 19 % in 2023 [1]. Since it can connect several sensors, smart objects, and everyday items by exchanging data, IoT has many applications in various scenarios. For instance, the industrial IoT [17] can help gather data from the machines so that humans do not need to reach or monitor them on-site. In the smart city scenario [76], IoT devices can support long-range transmission to gather information such as traffic data, temperature, and humidity. In general, cloud computing [81] improves the speed of message processing since it has strong computation ability. However, with the number of messages and the message size becoming enormous, the traditional cloud computing paradigm (e.g., Google GCP and Amazon AWS) faces the problem of long distance/high latency/large volume between the cloud platform and the IoT devices. In recent years, fog and edge computing have emerged to improve performance. Edge devices can communicate real-time data to the network to decrease transmission delay. Fog devices can further provide stronger computation ability than edge devices for complicated computation tasks.

1.1 Emerging Message Integrity Problem

During transmission, messages are susceptible to modification threats [77]. For example, once the data is changed in the IIoT scenario, the machine may conduct incorrect operations and cause great failure in the product. The emergency situation should be transmitted to the actor to handle in the wireless sensor and actor network [101]. The actor cannot accurately respond to the emergency if the message is falsified. A similar situation will also happen in the smart city scenario. The attack always exists in the network communication layer [96]. There are two solutions in cryptography: Message Authentication Code (MAC) [11] and Digital Signature [32]. The two kinds of schemes have different advantages. On one hand, the MAC can be built from cryptographic hash only and is thus computationally efficient. On the other hand, the digital signature supports some extra properties, such as undeniability [26]. Hence, they can be widely applied in IoT scenarios [28] with different requests.

However, our key observation is that the current MAC or signature schemes confront some troubles in practical IoT situations since they do not consider the computation ability of the individual IoT node. For example, in the Industrial IoT (IIoT) scenario, the sensor nodes are cheap, so their hardware has a limited workload. In the smart city, the nodes that gather traffic data are deployed in a large area. As a consequence, their hardware cannot be updated frequently. In both situations, when an individual IoT node confronts many messages, it may encounter difficulties completing the verification tasks since the sender generates MAC or signatures based on the whole message. To solve the problem, subset verification is necessary, which means it can verify a subset of the message and allocate the rest subset to a trusted third party.

Currently, there are some existing solutions. The locally sensitive hashing (LSH)

[50][94] can solve the problem to some degree. A similar document will be mapped to the same bucket with high probability. If the similarity of the message subset is smaller than the threshold of the LSH, the hashing results of the message subset are the same as the original message. However, the false positive and false negative problems in the LSH cannot be ignored. The server-aided signature [23] can also solve the computation bottleneck problem by outsourcing the computation task to a third party. However, the third party does not check the message integrity. Once the message has been modified by the adversary, the third party cannot figure it out until the IoT node finishes the verification. This process still wastes the computation ability of IoT nodes. In our assumption, a third party (e.g., fog node or edge node) should be involved in verification rather than computation. The IoT nodes can outsource a subset of the messages to a trusted third party for verification. If the third party find out the subset of the messages has been modified, the rest verification process will be terminated. Specifically, in the IIoT scenario, the distance between devices is short. One fog node can help gather and verify the message. On the contrary, in the smart city, the devices are mobile, and the distance between them is long. The edge computing paradigm is suitable.

Besides subset integrity, subset privacy is also important. Generally, the MAC only focuses on message integrity while ignoring the privacy problem [37]. That is, the message may contain useless (in terms of integrity) but sensitive content. A naive solution is that when a client asks for the message, he can delete the useless contents and send the rest to the client. However, different clients consider different content useless. For example, when a sensor node gathers temperature data from one room, one client needs the data during working hours to check how human activities affect temperature. The other client needs the data during the night for emergency and anomaly detection. Deleting message contents and re-generating MAC cannot be pre-processed in this scenario since the sensors do not know which subset the clients need before the message transmission. Merkle Hash Tree [67] can somewhat solve the subset privacy. However, the storage and the communication cost will be enormous. To solve this situation, we consider whether we can generate the MAC before transmission while packing the useless content so that the client cannot know it. In [46], **Prefix Verifiable Message Authentication Code (PMAC)** is proposed to protect both message integrity and privacy in Location-based Services (LBS) [29]. LBS provides location-aware services according to devices' geographical location, such as local weather services and location-based games. In these situations, the requests for location accuracy are different. For instance, the weather services only need to know the living city, whereas a location-based game requires a more precise location. However, the attacker can modify the location message [72][38]. For example, a taxi should not leave its operating area in some cities [97], while the attacker can modify the location message so the taxi can leave its operating area. However, PMAC has a limitation. When the message contains two or more kinds of information, and the client randomly asks for the content inside, PMAC cannot be applied. Consequently, subset privacy and subset integrity should be considered.

1.2 Fog and Edge Computing

Cloud computing enables individuals and corporations to access computing resources via the Internet on demand at high prices. It affords customers enhanced flexibility and scalability in contrast to conventional on-premises computing models. Nevertheless, a significant hurdle hindering the integration of cloud computing with IoT applications is the considerable distance, resulting in high transmission delays and large data volumes between the client and the cloud [59].

Therefore, fog and edge computing are proposed to solve the problem. Edge computing [48] is a computational paradigm designed to relocate computation and data storage nearer to data sources, thereby reducing transmission delays compared to cloud computing. In other words, edge computing is required to share the pressure of the cloud and take charge of tasks within its scope of the edge. Unlike centralized ownership by large corporations like Google, Amazon, and Huawei, edge devices are typically owned by individual users. Nevertheless, individual edge nodes do not have enough computation ability [86], which leads to the computation bottleneck. There are two solutions to address the issue. First, several edge nodes can cooperatively solve the computation tasks. Based on that, collaborative edge computing is proposed [68]. Each edge node can select a subset of the tasks to compute. Another solution is updating the edge node by increasing its computation ability. In other words, a device with significant computation ability and without the drawbacks of a cloud platform should be applied. Thus, the fog computing [41] is proposed. Similar to the cloud, fog provides data computing, storage, and application services to end users. In fog computing, facilities that can provide resources for services at the edge of the network are called fog nodes. Fog computing inherits the advantages of both cloud computing and edge computing. On one hand, the cost and transmission delay in fog computing are smaller than those in cloud computing. On the other hand, the fog node has stronger computation ability than the edge node. In summary, fog and edge computing can assist in solving the subset integrity problem.

1.3 Contribution and Outlines

In this paper, we want to solve the message subset integrity problem. Specifically, the subset integrity means that the verifier can check the correction of the message subset even if the corresponding MACs or signatures are generated based on the whole message. When one verifier confronts the computation bottleneck in verification, he can randomly select as many verification tasks as possible and offload the rest to other participants. Primarily, we introduce the subset privacy, which emerges as a subsidiary concern within the subset integrity paradigm in Chapter 3. When a participant solely necessitates access to a message subset, maintaining the confidentiality of the remaining subset becomes imperative. However, since a single message may be relevant to multiple participants with varying privacy requirements, generating distinct MACs for identical messages proves impractical. A scheme ensuring the immutability of integrity proofs despite modifications to privacy requests is necessary to address this challenge. Subsequently, we provide an empirical study on subset integrity with the assistance of fog computing in Chapter 4. Fog nodes, endowed with substantial computational resources, can carry large amounts of verification tasks. Consequently, the subset verification only needs two determined entities. In the meantime, we provide an optimization solution to decrease the energy consumption of the message receiver. Notably, the computation resources consumed increase proportionally with the number of subsets verified by a node. Therefore, it is essential to find an optimization solution to minimize the receiver's computation ability consumption. Finally, we extend the research to a general situation where multiple entities can cooperate to verify the message in Chapter 5. Since the edge node can be widely applied in the smart city, the number of participating edge nodes and their computation ability are unknown. Consequently, the signature should support several edge nodes to cooperatively complete the verification. Moreover, we figure out an optimization solution to minimize the energy consumption of the edge nodes.

In summary, the contributions of this thesis are listed as follows:

- **Scalable Prefix Verifiable Message Authentication Code (SPMAC).** We propose the SPMAC scheme to provide message integrity and subset privacy service. This scheme enables the sender to generate a single MAC for different receivers with varying privacy requirements. The SPMAC scheme can not only inherit the advantages of PMAC but also overcome its limitations.

- **Partial Verification Message Authentication Code (PV-MAC).** We propose PV-MAC for subset verification in a fog-based IIoT scenario. The MAC is generated based on the whole message, and the fog node can randomly choose cells for verification rather than being determined statically by the message sender.

- **Elliptic Curve based Multi-hop Sanitizable Signature (ECMSS).** We propose ECMSS to support subset verification among several entities in the smart city. Additionally, the ECMSS scheme can support subset aggregation. This capability enables edge nodes to concatenate incoming messages and aggregate signatures, even when different private signing keys generate the signatures. In the meantime, ECMSS permits the edge node to select a subset of the message for random verification.

The rest of this thesis is organized as follows:

- **Chapter 2.** We provide a systematic review of the related literature and preliminaries. Specifically, we discuss the privacy and integrity assurance schemes in cryptography, as well as the computation offloading schemes.

- **Chapter 3.** We introduce the sub-problem in subset integrity, called subset privacy. In the meantime, the message integrity should be guaranteed. To solve the problem, we propose a Scalable Prefix Verifiable Message Authentication Code (SPMAC). It is a novel and efficient scheme to concurrently solve message integrity and subset privacy.

- **Chapter 4.** We demonstrate the subset integrity problem between two entities. To solve the problem, we propose a Partial Verification Message Authentication Code (PV-MAC) in a fog-based IIoT scenario. In the meantime, we provide an optimization solution so that the fog node can carry as many verification tasks as possible to decrease the receiver's energy consumption. Furthermore, a Partial Verification Chained Message Authentication Code (PV-CMAC) is proposed to reduce communication costs.

- **Chapter 5.** In this chapter, we show a general situation, which is the subset integrity verification among multiple entities. To address the problem, we propose an Elliptic Curve Multi-hop Sanitizable Signature (ECMSS) in the smart city scenario. The ECMSS can support multiple edge nodes to verify the message cooperatively. Besides supporting subset verification, the scheme can support subset aggregation. Specifically, different signers can generate the signatures for different signatures in the smart city scenario. The aggregator can combine the signatures even if different keys generate them. Furthermore, an integer-based optimization solution is proposed to minimize the energy consumption among the edge nodes.
- **Chapter 6.** In this chapter, we conclude this thesis and show some possible research directions in the future.

Chapter 2

Literature Review and Preliminaries

2.1 Related Work

2.1.1 Message Integrity

To ensure the message integrity, signature and MAC are two standard solutions. Johnson, Menezes, and Vanstone proposed [52] the ECDSA scheme, a notable solution in IoT scenarios. Based on that, the identity-based signature [22] was introduced so that the key can be generated according to the identities. Furthermore, Zhiwei, Ruirui, and Shaohui provided [98] a server-aided signature for smart cards and wireless sensors by applying the attributed-based signature. Shahid and Abid [87] proposed Smart Digital Signatures with a novel key compression tree in the quantum era. Some existing solutions had some extra properties. For example, the homomorphic signature was proposed in [53]. Based on that, an identity-based homomorphic scheme is proposed for blockchain [66], and [64] applied a homomorphic signature in an IoT scenario, which can combine the incoming message. Camenisch, Maurer, and

Stadler [14] provided a Schnorr signature scheme, while [54] applied it to support multiple message signing and key aggregation. The keyed-hashing MAC (HMAC) was proposed for the MAC schemes in [55]. Based on that, Erroutbi, El Hanjri, and Sekkaki [31] applied HMAC in the IoT scenario. The MAC scheme also has extra properties. In [46], the authors proposed the Prefix-Verifiable Message Authentication Code to protect the message integrity as well as the location privacy. In [4], the authors introduced a stateful MAC scheme named Progressive MACs (ProMACs) in data stream authentication so that the verification can resynchronized once the data is missing. Broadbent and Wainewright [12] extended the MAC scheme in quantum security.

However, traditional integrity assurance schemes do not support subset integrity verification. To solve the problem, a sanitizable signature [7][60] allowed a delegator to modify parts of the messages. This scheme can be applied in several scenarios, such as the Industrial IoT [49] scenario. Furthermore, Samelin and Slamanig [83] provided a policy-based sanitizable signature scheme for a blockchain scenario. Apart from the sanitizable signature, some concepts were related to privacy-assurance signature. The redactable signature scheme(RSS) [13] has properties similar to those of a sanitizable signature. The RSS can delete a subset of the messages without knowing the private key. Specifically, there are some applications of RSS. The Pöhls and Samelin [78] provided an accountable RSS scheme that supports accountability. [63] proposed the RSS for cloud storage. However, these works do not consider the multi-hop scenario, which is common in smart city scenarios.

2.1.2 Message Privacy

Message privacy has many applications. In LBS security, users' location and trajectory are sensitive and should be kept secret. In a wireless sensor network, the data is transmitted among several sensor nodes. Data privacy should be guaranteed so that the adversary cannot know the data during the transmission [104]. In data publication, an attacker with solid background knowledge can obtain sensitive user information by comparing two or more publications. In IoT, a general idea is encryption. The encryption scheme can be divided into two kinds: symmetric encryption and asymmetric encryption. Furthermore, Gentry proposed fully homomorphic encryption, which means the client can compute the ciphertext without knowing the plaintext. Subsequently, the authors applied homomorphic encryption to the IoT scenario to ensure message privacy [75]. Besides, the traditional public-key encryption confronts a problem: significant storage of public keys in Public Key Infrastructure (PKI), Shamir provided the identity-based encryption (IBE) [89], and Cocks provided an IBE scheme based on quadratic residues[24]. Instead of encrypting messages by a public key, the receiver's identity was used to encrypt the messages. Thus, it can be widely applied in IoT scenarios to solve privacy problems [19].

Another solution is differential privacy (DP) and local differential privacy. By adding the noise to the data, the client cannot know the exact value of the data. For instance, the privacy of the trajectory is protected by omitting sensitive information or adding noise in [102] [93]. In [100], the authors combined DP with edge computing to protect message privacy in an IoT scenario. [105] applied LDP in temporal trajectory. The authors added the noise in the time order rather than the location data. In this situation, the efficiency is better than that of the traditional LDP mechanism.

2.1.3 Computation Offloading

Computation offloading is applied to collaborative edge computing and fog computing. The optimization problem attracts excellent attention. Generally, there are two kinds of scenarios. First, only several edge nodes cooperate to finish the tasks [69]. The other scenario is the hybrid mode, which means the network contains edge nodes, fog nodes, and cloud platforms [44]. Generally, there are several objectives. For instance,

[70] used game theory to find the Nash Equilibrium so that the time consumption declines. The authors in [92] used Multi-Armed Bandit to optimize the performance of multi-hop offloading. In [44], the objective was to concurrently minimize the system time and energy consumption.

Currently, most of the work concentrates on optimization problems in computation offloading. However, the subset verification is also important. Due to the limited computation ability, the verification process should be conducted among several third parties. The participant should undertake the computation tasks and check the verification result. This process is complex because all the participants independently choose the cells for computation, while there is only one signature for verification.

2.2 Preliminaries

2.2.1 Prefix-Verifiable Message Authentication Code

Prefix-verifiable message authentication code (PMAC) was proposed in [46]. The objective is to solve the problem how the authenticator can authenticate the message without disclosing the information that the verifier does not need to know. In this scheme, the data source (DS) only needs to generate one MAC for the message. The data user (DU) can then guarantee the integrity of the cells it ought to be aware of while is unable to acquire more knowledge than he should know. Table 2.1 shows the notations of PMAC, and Algorithm 1 shows how PMAC works.

Π is a character-wise associative, $\Pi(\mathbf{x}) = \prod_{i=1}^{N} \pi(x_i, i)$ and π is prime mapping function, which maps each cell to a distinct prime. $su(\mathbf{x})$ is the cells which are useless for the DU. In this scheme, the useless cells will be packed into a package σ, and prevent attackers or receivers from knowing them. In the meantime, the receiver can use σ to reconstruct the PMAC **without recomputing** it.

Table 2.1: Variables List for PMAC

Parameter	**Instruction**
$\mathbf{x}$	The message
α	The symmetric key for MAC generation
N	The number of cells in the message
$\mathbb{G}$	The multiplicative cyclic group
g	A random generator of the multiplicative cyclic group
p	Large prime for the multiplicative cyclic group
ψ_k	A keyed pseudorandom function
γ	Random nonce
t	The timestamps of message
π	Prime mapping
$su(\mathbf{x})$	The cells which are useless for the DU
$pre(\mathbf{x})$	The cells which are necessary for the DU

However, the PMAC still confronts some troubles when the disclosed cells are randomly placed. As shown in Figure 3.1(b), we generate MAC for the message. However, the useless cells are not adjacent to each other. We can separate the message into 2 pieces, the first piece is from x_1 to x_8, denoted by $\mathrm{ps}_1(\mathbf{x})$. The second piece is from x_9 to x_{16}, denoted by $\mathrm{ps}_2(\mathbf{x})$. Two pieces follow the prefix-suffix Verification model so that two suffixes can be constructed. Subsequently, we can send the processed message with the MAC to the verifier (i.e., $(x_1, x_2, x_3, \sigma_1, x_9, x_{10}, x_{11}, x_{12}, \sigma_2)$). After receiving it, the verifier can construct MAC_1 and MAC_2 for $\mathrm{ps}_1(\mathbf{x})$ and $\mathrm{ps}_2(\mathbf{x})$,

Algorithm 1 Framework of PMAC

```
Input: Symmetric key $(\alpha, k)$; random nonce $\gamma$; message $\mathbf{x}$ timestamp $t$
Output: PMAC
  DS, DU $\leftarrow \alpha$, DS $\leftarrow \gamma$
  PMAC $= g^{\alpha(\Pi(\mathbf{x})\gamma+\psi_k(t))} \bmod p$
Input: Suffix of the message $su(\mathbf{x})$; $\gamma$;
Output: The pack $\sigma$
  $\sigma = g^{\Pi(su(\mathbf{x}))\gamma} \bmod p$
Input: $(\alpha, k)$; prefix of the message $pre(\mathbf{x})$; timestamp $t$; $\sigma$;
Output: 0 or 1
  PMAC' $= \left(\sigma^{\Pi(pre(\mathbf{x}))} g^{\psi_k(t)}\right)^{\alpha}$
  if PMAC' == PMAC then
      return 1
  else
      return 0
  end if
```

$$\begin{aligned}
\mathrm{MAC}_1 &= \left(\sigma_1^{\Pi(pre_1(\mathbf{x}))} g^{\psi_k(t)}\right)^{\alpha} \\
&= g^{\alpha(\Pi(ps_1(\mathbf{x}))\gamma_1+\psi_k(t))} \bmod p \\
&= g^{\alpha(\Pi(ps_1(\mathbf{x}))\gamma_1)} \cdot g^{\alpha(\psi_k(t))} \bmod p \\
\mathrm{MAC}_2 &= \left(\sigma_2^{\Pi(pre_2(\mathbf{x}))} g^{\psi_k(t)}\right)^{\alpha} \\
&= g^{\alpha(\Pi(ps_2(\mathbf{x}))\gamma_2+\psi_k(t))} \bmod p \\
&= g^{\alpha(\Pi(ps_2(\mathbf{x}))\gamma_2)} \cdot g^{\alpha(\psi_k(t))} \bmod p \\
\mathrm{MAC} &= g^{\alpha(\Pi(\mathbf{x})\gamma+\psi_k(t))} \bmod p \\
&= g^{\alpha(\Pi(\mathbf{x})\gamma)} \cdot g^{\alpha(\psi_k(t))} \bmod p
\end{aligned} \tag{2.1}$$

Inside, $\gamma = \gamma_1 \cdot \gamma_2$ and $\mathrm{pre_i}(\mathbf{x})$ is the prefix of $\mathrm{ps_i}(\mathbf{x})$.

The verifier knows the symmetric key (α, k), however, he cannot know $\Pi\left(ps_1(\mathbf{x})\gamma_1\right)$ and $\Pi\left(ps_2(\mathbf{x})\gamma_2\right)$ due to the prime mapping and multiplicative cyclic group. We observe the difference among MAC$_1$, MAC$_2$, and MAC,

$$\begin{aligned} A &= g^{\alpha \Pi(ps_1(\mathbf{x}))\gamma_1} = \left(g^{\alpha}\right)^{\Pi(ps_1(\mathbf{x}))\gamma_1} \\ B &= g^{\alpha \Pi(ps_2(\mathbf{x}))\gamma_2} = \left(g^{\alpha}\right)^{\Pi(ps_2(\mathbf{x}))\gamma_2} \\ C &= g^{\alpha \Pi(ps_1(\mathbf{x}))\Pi(ps_2(\mathbf{x}))\gamma} = \left(g^{\alpha}\right)^{\Pi(ps_1(\mathbf{x}))\Pi(ps_2(\mathbf{x}))\gamma} \end{aligned} \tag{2.2}$$

We need to prove the verifier cannot compute C according to A and B. The proof consists of two kinds,

- It is difficult to indirectly compute C by computing the index of A and B in probabilistic polynomial time.
- It is difficult to directly compute C from A and B in probabilistic polynomial time.

Proof of difficulty in indirect computation. According to Discrete Logarithm Problem (DLP) [108], A, B, and C are indistinguishable under the chosen plaintext attacks. Therefore, we cannot compute the index of A and B.

Proof of difficulty of direct computation. Let us prove this by contradiction. If there were an algorithm $\mathcal{A}$ for the verifier to combine A and B to compute C, we can design an algorithm $\mathcal{A}'$ for attacking the Computational Diffie-Hellman problem (CDH). Recall the CDH problem is that, given a public base g, a large prime p, $A' = g^{a'} \bmod p$ and $B' = g^{b'} \bmod p$, to compute $C' = g^{a'b'} \bmod p$ without knowing a' and b'. For the problem that an attacker does not know a' and b', $\mathcal{A}'$ simply asks $\mathcal{A}$ to combine A and B to compute C with $g = g^{\alpha}$, $a' = \Pi\left(ps_1(\mathbf{x})\right)\gamma_1$ and $b' = \Pi\left(ps_2(\mathbf{x})\right)\gamma_2$. Then $C = C'$, which solves the CDH problem. This violates the assumption there is no probabilistic polynomial time-bounded algorithm for the CDH problem.

In summary, the limitation of PMAC is based on the multiplicative cyclic group. Therefore, we should replace it with other possible solutions so that the new PMAC can be suitable for the general situation.

2.2.2 Hierarchical Secret Sharing

Secret sharing [88][36] is an important cryptographic primitive. One dealer can distribute one secret among several participants, each of whom can get one shadow [30] for secret recovery.

Formally, in a (t, n) secret sharing, there are one dealer and n participants. The dealer can generate a polynomial $f(x) = s + \sum_{i=1}^{t-1} a_i x^i \bmod p$. p is a large prime, while a_i is random integer, which is smaller than p. Each participant (P_j) receives his shadow $(x_j, f(x_j))$ [84], where $f(x_j)$ is the subshare of the shadow. By applying Lagrange Interpolation [16], at least t authenticated participants can recover the secret (s). On the contrary, any $t-1$ or fewer authenticated or unauthenticated participants cannot recover the secret according to Eq. 2.3.

$$\begin{bmatrix} 1 & x_{i_1} & x_{i_1}^2 & \cdots & x_{i_1}^{t-1} \\ 1 & x_{i_2} & x_{i_2}^2 & \cdots & x_{i_2}^{t-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{i_t} & x_{i_t}^2 & \cdots & x_{i_t}^{t-1} \end{bmatrix} \begin{bmatrix} s \\ a_1 \\ \vdots \\ a_{t-1} \end{bmatrix} = \begin{bmatrix} f(x_{i_1}) \\ f(x_{i_2}) \\ \vdots \\ f(x_{i_t}) \end{bmatrix} \tag{2.3}$$

According to these properties, we can derive that

Theorem 1. *There is only one polynomial (e.g., $f(x)$) with $n-1$ degree, through which n shadows $(x_1, f(x_1))$,$(x_2, f(x_2))$...$(x_n, f(x_n))$ can interpolate.*

Proof. Suppose there is another different $n-1$ degree polynomial (e.g., $b(x)$), through which the same shadows can interpolate. Then we can compute $c(x) = f(x) - b(x)$, which has two properties,

- $c(x)$ is at most $n-1$ degree polynomial and non-zero polynomial.
- $c(x)$ has n root.

However, these two properties are in contradiction. Therefore, $f(x)$ and $b(x)$ are equal. □

In hierarchical secret sharing (HSS) [74], the dealer separates participants into different levels. Formally, we call it ($\mathbf{t}$, n) hierarchical secret sharing, where $\mathbf{t} = \{t_i\}_{i=1}^{L}$, which is the threshold in each level, and L is the number of levels.

Disjunctive [18] and conjunctive HSS [107] are the two types of HSS. The main difference is the relationship between the two adjacent levels. Given that there are two levels (L_1 and L_2) and the thresholds in each level are t_1 and t_2. The disjunctive HSS requires at least t_1 participants from L_1 **or** at least t_2 participants from L_1 and L_2 to recover the secret. On the contrary, the conjunctive HSS needs at least t_1 participants from L_1 **and** at least t_2 participants from L_1 and L_2. According to the two kinds of HSS, we can determine that the participants from the highest level **must** participate in recovering the secret. In other words, we can consider that different participants have different weights according to their levels. Once the sum of the weight exceeds the threshold weight of the whole message, the participants can recover the secret. There are two conditions to recover the master secret.

C1. In the highest level (L_1), the number of participants is not less than the threshold t_1.

C2. In each participated level L_i, the number of participants is not less than the threshold t_i. In the meantime, the participants belong to the previous level ($j < i$), and the number of participants is equal to $t_j - 1$

Example 1 illustrates how it works.

Example 1. *There are ten participants with two different levels (L_1 and L_2). The first level's threshold and number of participants are $t_1 = 3$ and $n_1 = 4$. The second level's threshold and number of participants are $t_2 = 4, n_2 = 6$. Specifically, P_1, P_2, P_3, P_4*

belong to first level and $P_5, P_6, P_7, P_8, P_9, P_{10}$ *belong to second level. Inside,* P_5 *is the header of the second level, which means he can also participate in secret recovery in the first level.*

Then the dealer can generate two polynomials in each level (e.g., $f_1(x) = 1 + 2x + 3x^2 \bmod 97$ *and* $f_2(x) = 86 + 4x + 6x^2 + 8x^3 \bmod 97$*).* 1 *is the master secret. Suppose the participants' IDs are* $1, 2, \ldots, 10$*, then the dealer can generate shadows for the participants according to their level and ID (e.g.,* $P_1 : (1, 6), \ldots, P_5 : (5, 11), P_6 : (6, 34), \ldots$*).*

When recovering the secret, either P_1*,* P_2*,* P_3 *or* P_1*,* P_2*,* P_5*,* P_6*,* P_7*,* P_8 *can recover it. Specifically,* P_1*,* P_2*,* P_3 *can recover the secret as traditional secret sharing, which satisfies* $C1$*. However, when* P_1*,* P_2*,* P_5*,* P_6*,* P_7*,* P_8 *are selected to recover the secret, the master secret recovery contains two steps.* P_5*,* P_6*,* P_7 *and* P_8 *first recover the secret, which is 86 in* $f_2(x)$*. Then the level header* P_5 *can obtain another subshare (e.g.,* $P'_5 : (5, 86)$*). Afterward,* P_1*,* P_2*, and* P'_5 *can recover the master secret, which satisfies* $C2$*.*

2.2.3 Chameleon Hashing

The hash function has been widely used since it has an important property called collision resistance. In other words, it is hard for the adversary to find two different inputs (e.g., $\mathbf{x}$ and $\mathbf{x}'$) so that their hash value are the same (e.g., $H(\mathbf{x}) = H(\mathbf{x}')$).

On the contrary, the Chameleon Hashing [56][51] is a trapdoor hash function, which allows a third party to find out the collision if he has a trapdoor key. However, it is still a collision-resistant hash function for the parties without the trapdoor key.

In [20], the authors provided a basic Chameleon Hashing scheme containing four steps.

- **System Parameter Generation**: It takes the security parameter λ, and outputs the parameters SP.

- **Key Generation**: Given a security parameter λ and SP, the key generation algorithm returns trapdoor/public key pairs (sk, pk).
- **Hash Computation**: It takes the public key pk, a message $\mathbf{x}$ and a random integer r as input, and outputs the hashed value $h = Hash(\mathbf{x}, r)$.
- **Collision Computation**: It takes the trapdoor key sk, a collision message $\mathbf{x}'$ and the hashed value h as input, and outputs r', which satisfying $h = Hash(\mathbf{x}, r) = Hash(\mathbf{x}', r')$.

There are two properties in a secure chameleon hashing.

- **Collision resistance**: Without the trapdoor key, there exists no efficient algorithm to find out another pair $(\mathbf{x}'', r'')$ so that $Hash(\mathbf{x}, r) = Hash(\mathbf{x}'', r'')$
- **Semantic security**: For all pairs of messages $\mathbf{x}$ and $\mathbf{x}'$, the distribution of the hashing results $Hash(\mathbf{x}, r)$ and $Hash(\mathbf{x}', r)$ are computationally indistinguishable.

The chameleon hashing can be used in several areas. One of the classic applications is in redactable blockchain [5]. Generally, the append-only blockchain is essential to protect the security of Bitcoin. However, an immutable ledger is not appropriate for all new applications that are being envisaged for the blockchain. Therefore, the chameleon hash can be applied to generate a redactable blockchain so that the participants can modify the content in specific circumstances.

2.2.4 Sanitizable Signature

Digital Signature is an important cryptographic primitive that can protect message integrity and authentication [45]. However, privacy should be considered in some scenarios. For instance, according to the Freedom of Information Act, the US government should release previously classified documents in 'sanitized' form. A delegator can generate a new signature for the sanitized message without knowing the key from the original signer and ensure the message integrity. To achieve that, the sanitizable signature [7] is proposed. Specifically, there are four steps.

- **Key Generation**: According to the security parameter λ, the signing and sanitization key pair are generated

$$(pk_{sign}, sk_{sign}), (pk_{sanit}, sk_{sanit}) \leftarrow 1^{\lambda}$$

- **Sign**: It takes the message $\mathbf{x}$, a private signing key sk_{sign}, a public sanitization key pk_{sanit} and the random integer r as input, and output the signature

$$\sigma \leftarrow Sign(\mathbf{x}, r, sk_{sign}, pk_{sanit})$$

- **Verify**: It takes the message $\mathbf{x}$, the public signing key pk_{sign}, a public sanitization key pk_{sanit}, the random integer r and the received signature σ as input, and outputs a bit b. If $b = 1$, the integrity of the processed message can be guaranteed.

$$b \leftarrow Verify(\mathbf{x}, pk_{sign}, pk_{sanit}, r, \sigma)$$

- **Sanitize**: It takes the original message $\mathbf{x}$, a substituted message $\mathbf{x}'$, a private sanitizing key sk_{sanit}, the public signing key pk_{sign} and the signature σ, and outputs a new signature σ'.

$$\sigma' \leftarrow Sanit(\mathbf{x}, \mathsf{x}', pk_{sign}, sk_{sanit}, \sigma)$$

The sanitizable signature has been used in various domains. One such example is content filtering. The language in the movie may not be suitable for the subscriber at a young age. In such cases, the platform or the administrator should replace offensive language with watered-down substitutes.

Chapter 3

Subset Privacy Problem in Subset Integrity

In this chapter, we begin to study the sub-problem in the subset integrity, which is called subset privacy. For instance, taxis can only work in certain areas in some countries. On one hand, the center should ask the driver to report their location periodically so that the driver cannot leave the area. On the other hand, the location information may disclose the driver's privacy (e.g., home address). A similar situation happens in the IoT scenario. For instance, when an examiner tries to verify the temperature data gathered by the sensors, he only needs to check a subset of the data (e.g., the data from working time). In the meantime, the sensor nodes cannot previously know which subset the examiner wants to check. Therefore, he should generate the MAC for all the data. Nevertheless, PMAC has a limitation. When there are two or more kinds of data, and the receiver randomly asks for the information inside, PMAC cannot be applied.

To address the problem, we introduce a Scalable Prefix Verifiable Message Authentication Code (SPMAC), which can support random message subset privacy. The SPMAC can be divided into two versions based on different mapping functions: Fibo-

Table 3.1: Comparison among The Three Schemes

	Mapping Function	Operation
PMAC	Prime Mapping	Multiplication
Fibo-SPMAC	Fibonacci Number Mapping	Addition
ECP-SPMAC	Elliptic Curve Point Mapping	Addition

SPMAC and ECP-SPMAC. Table 3.1 compares the three schemes in mapping functions and combination operation.

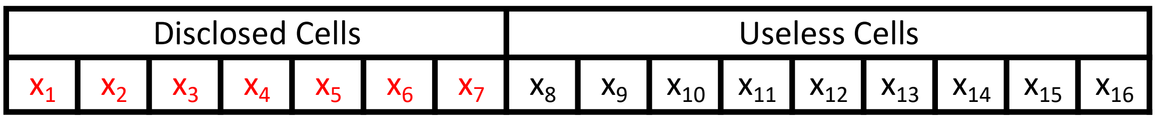
In summary, the contribution in this chapter is as follows,

1. We propose the SPMAC scheme to solve the subset privacy problem, which not only inherits the advantages of PMAC but also overcomes the limitations of PMAC.

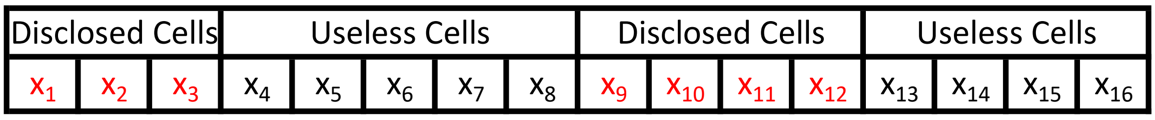
2. We propose two concrete SPMAC instantiations according to different mapping functions, and each of them can be applied in various practical situations.

3. We prove that the SPMAC scheme performs better in time consumption and communication cost than PMAC.

The remainder of this chapter is organized as follows. Chapter 3.1 formally defines our problem. Chapter 3.2 introduces the Scalable Prefix Verifiable Message Authentication Code (SPMAC). Chapter 3.3 analyzes the security of SPMAC, and Chapter 3.4 provides the experimental results comparing SPMAC with some existing schemes. Chapter 3.5 gives a summary of this chapter.



(a) Prefix-suffix Verification Model



(b) Random Verification Model

Figure 3.1: The Disclosed Cell Verification Model

3.1 Problem Definition

In this chapter, we examine how a user authenticates his messages by disclosing the minimum cells. Specifically, the message ($\mathbf{x} = (x_1, x_2 ... x_n)$) contains several cells (i.e., Figure 3.1), and only some cells, called the **disclosed cell** (i.e., the cells in red), are needed by other people. The rest of the cells called the **useless cell** (i.e., the cells in black), should be kept private. Moreover, the disclosed and useless cell placement like Figure 3.1(a) is called **Prefix-Suffix Verification Model**. The placement in Figure 3.1(b) is called **Random Verification Model**.

Given that various receivers can ask for different cells to be exposed, it would be inefficient to generate MAC for each request on the fly, leading to considerable delays and resource wastage. As a result, the sender can generate the MAC prior to any transmission. For instance, a location-based game needs a more exact location, whereas the local weather services merely need the user's city. It wastes computation resources to generate MAC for both services on the fly. In such a case, the sender cannot change the placement of each cell or separate the message into several pieces with a small size. Therefore, our proposed scheme should satisfy subset privacy, which can be expressed as Eq. 3.1

$$\mathrm{MAC}(\mathbf{x}) = \mathrm{MAC}(||_{i=1}^{s}(pre_i(\mathbf{x}), T(su_i(\mathbf{x}))))$$

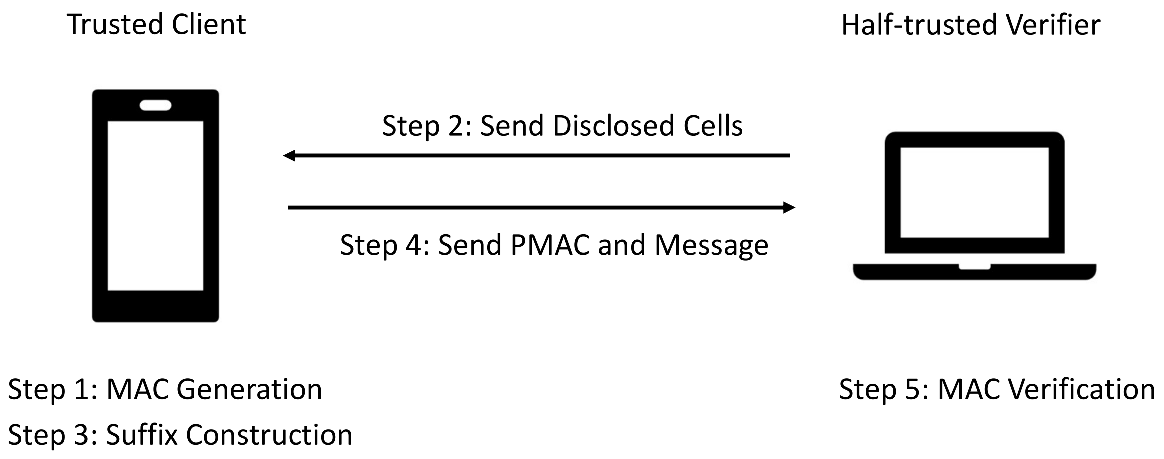

Trusted Client
Half-trusted Verifier
Step 2: Send Disclosed Cells
Step 4: Send PMAC and Message
Step 1: MAC Generation
Step 3: Suffix Construction
Step 5: MAC Verification

Figure 3.2: The System of PMAC

According to the disclosed cells, the message can be divided into s (prefix, suffix) pairs. $(pre_i(\mathbf{x}), T(su_i(\mathbf{x}))$ is the i-th pair. $||()$ is the concatenating function, which combines the (prefix, suffix) pairs like Eq. 3.1. T means packing the suffix into an undecryptable package.

$$(pre_1(\mathbf{x}), T(su_1(\mathbf{x})), \ldots, pre_s(\mathbf{x}), T(su_s(\mathbf{x}))) \tag{3.1}$$

There are two participants in the system, according to Figure 3.2, a trusted client and an honest-but-curious verifier. The client sends several messages to the verifier through a public channel. The honest-but-curious verifier is a receiver who only needs parts of the message from the client. However, he may try to learn extra information from the message (e.g., the value of useless cells). Specifically, the client generates the MAC for the message. Then, according to the disclosed cells, the client constructs the suffixes and transmits the processed message to the verifier. After receiving the processed message, the verifier regenerates the MAC according to the prefixes and the suffixes. Simplified speaking, there is an interaction between the client and the verifier. The verifier tells which cells he wants, and then the verifier can return the processed message with the MAC.

As much, the message transmission model confronts two security threats.

1. Message integrity: The attacker cannot alter the cells without changing the MAC.
2. Message privacy: The verifier can only learn the disclosed cells.

We give the definition of the adversary model in this system as follows. During the message transmission, the attacker may try to modify the message so that he can counterfeit the message and deceive the verifier. In the meantime, the verifier only needs the disclosed cells. However, he may try to learn the useless cells when he receives the processed message.

3.2 Scalable Prefix Verifiable Message Authentication Code

In this section, we introduce the scalable PMAC schemes. Generally, SPMAC schemes overcome the drawbacks of PMAC, which is explained in the chapter. 2.2.1. We apply the elliptic curve cyclic group in the SPMAC scheme. The combination operation is addition. There are two different mapping functions, which map each cell to a distinct value. The first one is based on Fibonacci mapping (Fibo-SPMAC), and the second one is based on elliptic curve point mapping (ECP-SPMAC).

Table 3.2 summarizes some notations in Fibo-SPMAC and ECP-SPMAC. Since both schemes are improvements to PMAC, they follow the same properties as PMAC.

- **Only the disclosed cells** need to be queried without exposing the whole message.
- **Only one MAC** needs to be verified when the requests for disclosed cells are different.

Table 3.2: Variables List for Fibo-SPMAC and ECP-SPMAC

Parameter	**Instruction**
a, b, p	The parameters of elliptic curve
$\mathbb{G}$	An elliptic curve cyclic group
g	A random generator belongs to $\mathbb{G}$
k	Secret key in message authentication code
Π	Character-wise associative
n	The number of cells in one message
γ	Random nonce
Fibo()	Fibonacci mapping function
ECP()	Elliptic Curve Point mapping function

3.2.1 Syntax of Fibo-SPMAC

In this part, we show the syntax of SPMAC. First, we show how SPMAC works with Algorithm 2. Then, we use a concrete example and Figure 3.3 to illustrate the SPMAC.

In an elliptic curve cyclic group, the basic combination operation is addition and scalar multiplication. Thus, we can map each cell to a distinct number according to its value and placement and combine the mapping result with an addition operation (e.g., $\sum_{i=1}^{n}(x_i \cdot g)$). Even if we map each cell to a distinct prime, it is easy to find the collision. The adversary can compute $\sum_{i=1}^{n}(x_i \cdot g) = (\sum_{i=1}^{n} x_i) \cdot g$. Afterward, the addition on the elliptic curve returns to the integer addition operation.

There is a critical theorem called **Zeckendorf's theorem** [43]. It means that every positive integer can be represented uniquely as the sum of one or more **distinct** Fibonacci numbers in such a way that the sum does **not** include any two **consecutive** Fibonacci numbers (e.g., $64 = 55 + 8 + 1$). In the SPMAC scheme, our solution is mapping each cell to a distinct and non-consecutive Fibonacci number (Fibo-mapping).

1. **Setup**(1^λ): The Setup algorithm is a probabilistic algorithm that takes as input a security parameter λ. It generates the symmetric key k, and a generator (g).
2. **Gen**$(\mathbf{x}, k, g, \gamma)$: The MAC algorithm takes as input the message $\mathbf{x}$, k, g, and a random nonce γ. It generates the SPMAC.
3. **SuffixGen**$(pre(\mathbf{x}), \gamma, g)$: The SuffixGen algorithm takes prefix $pre(\mathbf{x})$ and γ as input, and outputs the suffixes $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_s)$.
4. **Verify**$(pre(\mathbf{x}),\ k,\ g,\ \Sigma,\ \text{SPMAC})$: The Verify algorithm takes $pre(\mathbf{x})$, k, g, Σ and SPMAC as input. It outputs a bit b. If $b = 1$, the $pre(\mathbf{x})$ has not been modified.

There are three main steps: SPMAC generation, suffix construction, and SPMAC verification.

First, the client generates the SPMAC according to Eq. 3.2

$$\text{SPMAC} = k \cdot (\Pi(\mathbf{x}) + \gamma) \cdot g \tag{3.2}$$

$\Pi(\mathbf{x}) = \sum_{i=1}^{n} \text{Fibo}(x_i, i)$, which means map each cell into a distinct Fibonacci number according to its value and placement. Subsequently, the verifier provides the disclosed cells and sends them to the client. When receiving the request, the client constructs the suffixes according to Eq. 3.3 so that the verifier cannot know the cells inside. When there are a number of verifiers who have different disclosed cells, the client only needs to repeat the suffix construction algorithms without recomputing the MAC.

$$\sigma_i = (\Pi(su_i(\mathbf{x})) + \gamma_i) \cdot g \tag{3.3}$$

$\sum_{i=1}^{s} \gamma_i = \gamma$, and s is the number of (prefix, suffix) pairs.

Algorithm 2 Framework of Fibo-PMAC

```
Input: Symmetric key $k$; random nonce $\gamma \in Z_p$; message $\mathbf{x}$; generator $g$
Output: SPMAC
 1: Client, verifier $\leftarrow k$
 2: Client $\leftarrow \gamma \in Z_p$
 3: SPMAC $= Gen(k, \mathbf{x}, \gamma, g)$
Input: Disclosed cells in the message $pre(\mathbf{x})$; $\gamma$; $k$; $g$ message $\mathbf{x}$;
Output: $\Sigma$
 4: Client $\leftarrow pre(\mathbf{x})$
 5: Client separates the message into several parts, which follow the prefix-suffix
    model. Then, he calculates the number of (prefix, suffix) pairs ($s$).
 6: for $i = 1$; $i < s$; $i++$ do
 7:     Client $\leftarrow \gamma_i$
 8:     Client $\leftarrow su_i(\mathbf{x})$
 9:     $\sigma_i = SuffixGen(k, su_i(\mathbf{x}), \gamma_i, g)$
10: end for
11: Verifier $\leftarrow \Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_s\}, SPMAC$
Input: $pre(\mathbf{x})$; $\sigma_i$; $k$; generator
Output: SPMAC'
12: SPMAC' $= Verify(pre(\mathbf{x}), \sum_{i=1}^{s} \sigma_i, k, g)$
13: if SPMAC == SPMAC' then
14:     return 1
15: else
16:     return 0
17: end if
```

After constructing the suffixes, the processed message follows like Eq. 3.4 if the message follows Fig 3.1(b).

$$(x_1, x_2, x_3, \sigma_1, x_9, x_{10}, x_{11}, x_{12}, \sigma_2) \tag{3.4}$$

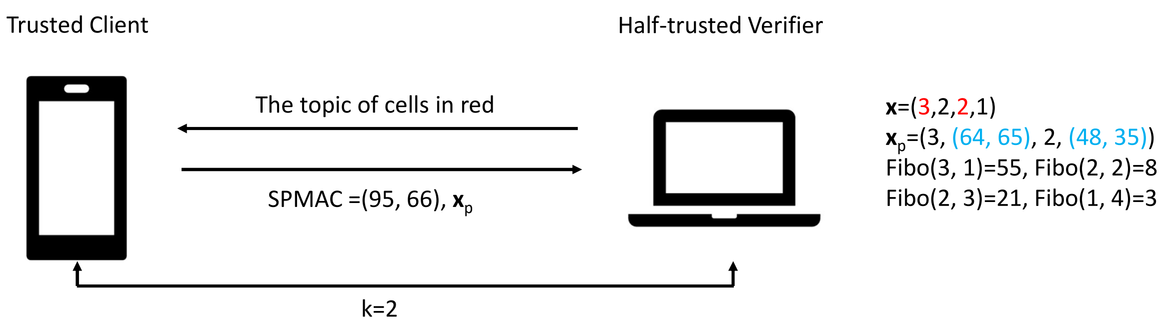
Subsequently, the client sends the processed message with the SPMAC to the verifier. Once the verifier receives the SPMAC, he computes $\sigma = \sum_{i=1}^{s} \sigma_i$ and then checks whether

$$k \cdot \sigma + \left(k \sum_{i=1}^{s} \Pi\left(pre_i\left(\mathbf{x}\right)\right) \right) g = k \cdot \left(\Pi\left(\mathbf{x}\right) + \gamma\right) \cdot g \tag{3.5}$$

Specifically, we use a concrete example to further explain how SPMAC works.

Example 2. *Suppose an elliptic curve follows* $E_{97}(1,1)$ *and* $g = (45, 56)$*. The message contains four cells (*$(3, 2, 2, 1)$*). The client maps each cell to a distinct and non-adjacent Fibonacci number (i.e.,* $\mathrm{Fibo}(2, 2) = 8$ *and* $\mathrm{Fibo}(2, 3) = 21$*). After that, the client combines the mapping results by addition and generates* $\mathrm{SPMAC} = (95, 66)$ *according to the elliptic curve cyclic group, a symmetric key (*$k = 2$*), and a random nonce (*$\gamma = 2$*) before the verifier asks for the message. When the client knows the disclosed cells (i.e., 3 and 2 in red) from the verifier, he can pack the useless cells (i.e., 2 and 1 not in red) and construct two suffixes (i.e.,* $(64, 65)$ *and* $(48, 35)$ *in blue) according to character-wise associative and the elliptic curve cyclic group.*

The verifier cannot find the cells in the suffixes according to the public-known factors and disclosed cells so that the anonymity of useless cells can be guaranteed. After the client sends the processed message $\boldsymbol{x}_p$ $(3, (64, 65), 2, (48, 35))$ *with the SPMAC to the verifier, he can combine the suffixes by addition, and reconstruct the SPMAC. After that, the verifier can check whether the message is modified so that message integrity can be ensured.*


Trusted Client
Half-trusted Verifier
The topic of cells in red
SPMAC =(95, 66), $\mathbf{x}_p$
k=2
$\mathbf{x}$=(3,2,2,1)
$\mathbf{x}_p$=(3, (64, 65), 2, (48, 35))
Fibo(3, 1)=55, Fibo(2, 2)=8
Fibo(2, 3)=21, Fibo(1, 4)=3

Client generates SPMAC 2×(55+8+21+3+2)×(45, 66) mod 97=(95, 66)
Client constructs suffix σ_1=(8+1)×(45, 66) mod 97=(64, 65)
σ_2=(3+1)×(45, 66) mod 97=(48, 35)
Verifier checks SPMAC 2×((64, 65)+(48, 35))+2×(55+21)×(45, 66) mod 97=(95, 66)

Figure 3.3: Concrete Example of SPMAC

3.2.2 Syntax of ECP-SPMAC

In this subsection, we introduce Scalable Prefix Verifiable Message Authentication Code on Elliptic Curve Point Mapping. In this scheme, the character-wise associative consists of **elliptic curve point mapping** and **addition**, and the security is based on **elliptic curve cyclic group**.

In this scheme, we map the plaintext on the given elliptic curve. We call it Elliptic Curve Point Mapping (ECP-mapping), which is based on Koblitz encoding [10]. Simply speaking, the point on an elliptic cyclic group is $(x, \sqrt{x^3 + ax + b} \bmod p)$, which is finding the quadratic residue of $x^3 + ax + b$. Therefore, the Koblitz encoding can be considered. Algorithm 3 shows the algorithm of Koblitz encoding.

ECP mapping consumes much time since both the client and the verifier need to compute the quadratic residue. Consequently, the client can generate a mapping look-up table like Table 3.3 for the client and verifier so that they can spend less time on mapping. The mapping look-up table can be generated and transmitted before SPMAC generation. Therefore, it is suitable for when the bandwidth is large.

First, we show how ECP-SPMAC works with Algorithm 4. Then, we use a concrete

Algorithm 3 Framework of Koblitz Encoding

```
Input: Initial key $(k_1, k_2)$; Elliptic curve $E_p$; plaintext $v$
Output: $(x, y)$
1: $i = 0$
2: while $E_p(k_1 \cdot v + i \cdot k_2)$ is not a quadratic residue do
3:     $i++$
4: end while
5: $x = k_1 \cdot v + i \cdot k_2$, $y = E_p(k_1 \cdot v + i \cdot k_2)$
```

Table 3.3: ECP-Mapping Look-up Table

value \ position	1	2	3	...
1	(x_{11}, y_{11})	(x_{12}, y_{12})	(x_{13}, y_{13})	...
2	(x_{21}, y_{21})	(x_{22}, y_{22})	(x_{23}, y_{23})	...
3	(x_{31}, y_{31})	(x_{32}, y_{32})	(x_{33}, y_{33})	...
...	...	...	...	...

example and Figure 3.4 to further illustrate it.

1. **Setup**(1^λ): The Setup algorithm is a probabilistic algorithm that takes as input a security parameter λ. It generates the symmetric key k.

2. **Gen**$(\mathbf{x}, k, \gamma)$: The MAC algorithm takes as input the message $\mathbf{x}$, k and a random nonce γ. It generates the SPMAC.

3. **SuffixGen**$(pre(\mathbf{x}), su(\mathbf{x}), \gamma)$: The SuffixGen algorithm takes prefix $pre(\mathbf{x}$ and γ as input, and outputs the suffixes $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_s)$.

4. **Verify**$(pre(\mathbf{x})$, k, Σ, SPMAC$)$: The Verify algorithm takes $pre(\mathbf{x})$, k, Σ, and SPMAC as input. It outputs a bit b. If $b = 1$, the $pre(\mathbf{x})$ has not been modified.

Algorithm 4 Framework of ECP-PMAC

```
Input: Symmetric key $k$; Initial key $k_1, k_2$; Random nonce $\gamma \in Z_p$; message $\mathbf{x}$
Output: SPMAC
1: Client, Verifier $\leftarrow k$
2: Client $\leftarrow \gamma \in Z_p$
3: SPMAC = $Gen(k, \mathbf{x}, \gamma)$
Input: Disclosed cells in the message $pre(\mathbf{x})$; $\gamma$; $k$ message $\mathbf{x}$;
Output: $\Sigma$
4: Client $\leftarrow pre(\mathbf{x})$
5: Client separates the message into several parts, which follow the prefix-suffix
   model. Then, he calculates the number of (prefix, suffix) pairs ($s$).
6: for $i = 1$; $i < s$; $i++$ do
7:     Client $\leftarrow \gamma_i$
8:     Client $\leftarrow su_i(\mathbf{x})$
9:     $\sigma_i = SuffixGen(k, pre_i(\mathbf{x}), su_i(\mathbf{x}), \gamma_i)$
10: end for
11: Verifier $\leftarrow \Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_s\}, SPMAC$
Input: $pre(\mathbf{x})$; $\sigma_i$; $k$
Output: SPMAC'
12: SPMAC' = $Verify(pre(\mathbf{x}), \sum_{i=1}^{s} \sigma_i, k)$
13: if SPMAC == SPMAC' then
14:     return 1
15: else
16:     return 0
17: end if
```

First, the client generates the SPMAC according to Eq. 3.6.

$$\text{SPMAC} = k \cdot (\Pi(\mathbf{x}) \cdot \gamma) \tag{3.6}$$

Insides, $\Pi(\mathbf{x}) = \sum_{i=1}^{n} \text{ECP}(x_i, i)$. After that, the verifier provides the disclosed cells and sends them to the client. After receiving the request, the client constructs the suffixes according to Eq. 3.7 so that the verifier cannot know the useless cells. When there are a number of verifiers who have different disclosed cells, the client only needs to repeat the suffix construction algorithms without recomputing the MAC.

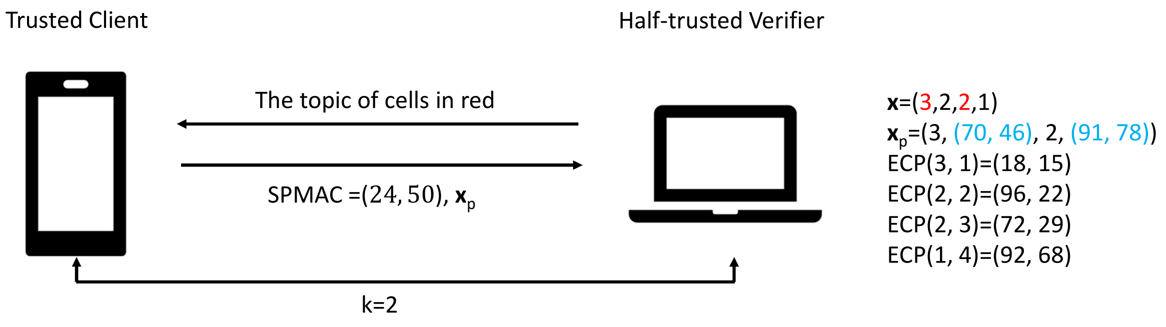
$$\sigma_i = (\Pi(su_i(\mathbf{x}))\gamma + \Pi(pre_i(\mathbf{x}))(\gamma - 1)) \tag{3.7}$$

After the steps, the processed message is like Eq. 3.4 if the message follows Fig 3.1(b). Once the verifier receives the SPMAC, he computes $\sigma = \sum\limits_{i=1}^{s} \sigma_i$ and then checks whether

$$k \cdot \sigma + \left(k \cdot \sum_{i=1}^{s} \Pi(pre_i(\mathbf{x}))\right) = k \cdot (\Pi(\mathbf{x}) \cdot \gamma) \tag{3.8}$$

Specifically, we use a concrete example to further explain how ECP-SPMAC works.

Example 3. *Suppose the elliptic curve follows* $E_{97}(1,1)$. *The message contains four cells* $(3,2,2,1)$. *The client maps each cell to a distinct elliptic curve point (i.e.,* $\text{ECP}(2,2) = (96,22)$ *and* $\text{ECP}(2,3) = (72,29)$*). After that, the client combines the mapping results by addition and applies an elliptic curve cyclic group, a symmetric key (*$k = 2$*), and a random nonce (*$\gamma = 2$*) to generate a* SPMAC $= (24,50)$ *before the verifier asks for the message. When the client knows the disclosed cells (i.e., 3 and 2 in red) from the verifier, the client can pack the useless cells (i.e., 2 and 1 not in red) and construct two suffixes (i.e.,* $(70,46)$ *and* $(91,78)$ *in blue) according to the elliptic curve cyclic group and character-wise associative. After that, the processed message is constructed. The verifier cannot find the cells in the suffix according to*



Client generates SPMAC $2 \times \big((18,15) + (96,22) + (72,29) + (92,68)\big) \times 2 \; mod \; 97 = (24,50)$

Client constructs suffix $\sigma_1 = \big((96,22) \times 2 + (18,15)\big) \; mod \; 97 = (70,46)$

$\sigma_2 = \big((92,68) \times 2 + (72,29)\big) \; mod \; 97 = (91,78)$

Verifier checks SPMAC $2 \times \big((70,46) + (91,78)\big) + 2 \times \big((18,15) + (72,29)\big) \; mod \; 97 = (24,50)$

Figure 3.4: Concrete Example of ECP-SPMAC

the public-known factors and disclosed cells so that the privacy of useless cells can be guaranteed. After the client sends the processed message $\boldsymbol{x}_p$ $(3,(70,46),2,(91,78))$ *with the SPMAC to the verifier, the verifier can combine the suffixes by addition, and regenerate the MAC. Afterward, the verifier can check whether the message is modified to ensure message integrity.*

3.3 Theoretical Analysis

In this section, we discuss the security of Fibo-SPMAC and ECP-SPMAC, as we mentioned in Chapter 3.1, which are message privacy and message integrity. We consider that a probabilistic polynomial time (PPT) attacker is involved. Precisely, the attacker can execute any PPT algorithm as an attack to accomplish his objective. Without altering the MAC, the PPT attacker attempts to falsify the message. In the meantime, the honest-but-curious verifier may try to learn the useless cells.

3.3.1 Message Subset Privacy

We prove that honest-but-curious verifier cannot know the $\mathbf{x}$ according to each $su_i(\mathbf{x})$ and public parameters. The privacy protection in both Fibo-SPMAC and ECP-SPMAC follows the same rule. Recall that all points on the elliptic curve construct a cyclic group($\mathbb{G}$).

Lemma 1. *For any random $r \leftarrow [0, |\mathbb{G}|)$, $r \cdot g$ has equal probability of being any element in $\mathbb{G}$. Formally, for any $\hat{g} \in \mathbb{G}$,*

$$\Pr\left[r \cdot g = \hat{g}\right] = 1/|\mathbb{G}|$$

Proof. We can derive that $\Pr\left[r \cdot g = \hat{g}\right] = \Pr\left[r = \hat{g}/g\right]$. Since r is randomly selected, the probability of r being a fixed element $\hat{g}/g$ is equal to $1/|\mathbb{G}|$. However, since division operation does not exist in an elliptic curve cyclic group, this process cannot be computed. □

The verifier can only observe the $\sigma_i = (\Pi(su_i(\mathbf{x})) + \gamma_i) \cdot g$ in Fibo-SPMAC or $\sigma_i = \Pi\left(su_i(\mathbf{x})\right)\gamma + \Pi\left(pre_i(\mathbf{x})\right)(\gamma - 1)$ in ECP-SPMAC. We can prove that the message subset privacy can be guaranteed.

Theorem 2. *The suffixes in the Fibo-SPMAC are indistinguishable according to the public information.*

Proof. According to Lemma 1, $\sigma_i = (\Pi(su_i(\mathbf{x})) + \gamma_i) \cdot g$ have equal probability of being any cells in $\mathbb{G}$ since γ_i is randomly selected and secret for the verifier. As a consequence, the verifier cannot learn anything about $\Pi(su_i(\mathbf{x}))$ and $su_i(\mathbf{x})$. □

Theorem 3. *The suffixes in ECP-SPMAC are indistinguishable according to the public information.*

Proof. Although there is no generator in Eq. 3.7, we can set $\Pi\left(su_i(\mathbf{x})\right)$ and $\Pi\left(pre_i(\mathbf{x})\right)$ as a generator. According to Lemma 1, $\Pi\left(su_i(\mathbf{x})\right)\gamma$ and $\Pi\left(pre_i(\mathbf{x})\right)(\gamma - 1)$ have equal probability of being any cells in $\mathbb{G}$. As a consequence, the verifier cannot learn anything about $\Pi(su_i(\mathbf{x}))$ and $su_i(\mathbf{x})$. $\square$

3.3.2 Message Integrity

In this part, we prove both SPMAC schemes are unforgeable. First, we introduce the unforgeability of Fibo-SPMAC.

Definition 1. ***(Unforgeability of Fibo-SPMAC)****. ECMSS(Setup, Sign, Verify) is unforgeable under adaptive chosen attacks if for any efficient algorithm $\mathcal{A}$ that the experiment $Unforgeability_{\mathcal{A}}^{Fibo\text{-}SPMAC}$ evaluates to 1 is negligible.*

Experiment $Unforgeability_{\mathcal{A}}^{SPMAC}$
-$(pk_{sign}, sk_{sign}) \leftarrow Setup(1^\lambda)$
-$SPMAC \leftarrow \mathcal{A}^{Sign(sk_{sign},\dot{})}(pk_{sign})$
-for $i = 1, 2, ..., q$, *denoted by* $SPMAC_i$, *the queries to the*
oracle$(\mathcal{O})$ *Sign return 1 iff Verify*$(pk_{sign}, SPMAC) = 1$

Specifically, $\mathcal{A}$ generates a random key from $Setup(1^\lambda)$. Then $\mathcal{A}$ submits MAC queries by sending different messages $(\mathbf{x}_i)$ and receives SPMAC$_i$ from the Oracle $(\mathcal{O})$. $\mathcal{A}$ wins if it can modify the messages by executing a polynomial time algorithm while leaving the MACs unchanged.

Theorem 4. *The Fibo-SPMAC is unforgeable in the random oracle model.*

Proof. For ease of presentation, let $m = (\Pi(\mathbf{x}) + \gamma)g$, the SPMAC can be expressed as:

$$\text{SPMAC} = k \cdot m$$

Let us prove this by contradiction. If there were a probabilistic polynomial time-bounded algorithm $\mathcal{A}$ to forge m and SPMAC, an algorithm $\mathcal{A}'$ can be designed to solve the Zeckendorf's theorem. Specifically, we can find two groups of Fibonacci numbers so that

$$\sum_{\mathrm{i}=1}^{n} Fibo\left(x_i\right) = \sum_{\mathrm{i}=1}^{n} Fibo\left(u_i\right) \quad \forall i, j \quad x_i \neq u_j$$

$\mathcal{A}'$ simply asks $\mathcal{A}$ to forge m with each $x_i = u_i$. Then a faked SPMAC' can be computed and satisfy SPMAC' = SPMAC. However, this violates Zeckendorf's theorem, which means each positive number can only be expressed as one group of distinct and non-adjacent Fibonacci numbers. □

Subsequently, we begin to introduce the unforgeability of ECP-SPMAC.

Definition 2. ***(Unforgeability of ECP-SPMAC).*** *ECMSS(Setup, Sign, Verify) is unforgeable under adaptive chosen attacks if for any efficient algorithm $\mathcal{A}$ that the experiment Unforgeability$_{\mathcal{A}}^{\textit{Fibo-SPMAC}}$ evaluates to 1 is negligible.*

Experiment Unforgeability$_{\mathcal{A}}^{SPMAC}$

-$(pk_{sign}, sk_{sign}) \leftarrow Setup(1^{\lambda})$

-$SPMAC \leftarrow \mathcal{A}^{Sign(sk_{sign},\cdot)}(pk_{sign})$

-for $i = 1, 2, ..., q$, denoted by $SPMAC_i$, the queries to the oracle($\mathcal{O}$) Sign return 1 iff Verify($pk_{sign}, SPMAC$) = 1

Specifically, $\mathcal{A}$ generates a random key from $Setup(1^{\lambda})$. Then $\mathcal{A}$ submits MAC queries by sending different messages ($\mathbf{x}_i$) and receives SPMAC$_i$ from the Oracle ($\mathcal{O}$). $\mathcal{A}$ wins if it can modify the messages by executing a polynomial time algorithm while leaving the MACs unchanged.

Theorem 5. *The ECP-SPMAC is unforgeable in the random oracle model.*

Proof. For ease of presentation, let $m = \Pi\left(\mathbf{x}\right)\gamma$, the SPMAC can be expressed as:

$$\mathrm{SPMAC} = k \cdot m$$

Let us prove this by contradiction. If there were a probabilistic polynomial time bounded algorithm $\mathcal{A}$ to forge m and SPMAC, an algorithm $\mathcal{A}'$ can be designed to solve the elliptic curve discrete logarithm problem (ECDLP). Specifically, the addition of two points on an elliptic curve yields a third point on the elliptic curve whose location has no immediately apparent relationship to the locations of the first two, and repeating the addition process many times over yields that may be essentially anywhere.

$\mathcal{A}'$ simply asks $\mathcal{A}$ to forge m by finding out two groups of elliptic curve points as follows:

$$\sum_{\mathrm{i}=1}^{n} ECP\left(x_i\right) = \sum_{\mathrm{i}=1}^{n} ECP\left(u_i\right) \quad \forall i,j \quad x_i \neq u_j$$

Then a faked SPMAC' can be computed and satisfy SPMAC' = SPMAC, which solves the ECDLP. This contradicts the assumption that there is no probabilistic polynomial time-bounded algorithm for ECDLP. □

3.4 Experiments

In this section, we use experimental results to show the advantages of Fibo-SPMAC and ECP-SPMAC as well as their applications.

There are four experiments. The first one focuses on the performance of SPMAC, PMAC, and HMAC [55] on different IoT devices. The second experiment is about the performances among SPMAC, PMAC, and HMAC in the prefix-suffix verification Model. We measure the CPU time for **cells mapping**, **MAC generation**, **suffix construction**, and **MAC verification**. Moreover, suffix construction and MAC verification contain the mapping function. Since the number of disclosed cells affects the length of prefixes and suffixes, the client and verifier should recompute the mapping function. To the best of our knowledge, PMAC is a state-of-the-art work that focuses on integrity assurance and privacy preservation. Therefore, it is essential to make a comparison with it. In the experiments, we apply HMAC for each cell and concatenate them together. The third experiment is about the comparison between Fibo-SPMAC and ECP-SPMAC in a random verification Model. The final one shows the practical situation. Considering the time for mapping in ECP-PMAC is larger than that in Fibo-PMAC, we can generate a mapping look-up table so that the verifier does not need to repeat the mapping function.

To evaluate the performance of Fibo-SPMAC and ECP-SPMAC in a real-life setting, we use the dataset, 'Young People Survey', which is provided on **Kaggle**. The dataset is about the preferences and habits collected by students of the Statistics class at FSEV UK, and there are 60 cells inside.

The prime, key, generator, and nonce for PMAC are 1024 bit, and the pseudo-random function adopts AES-256 in PMAC. The elliptic curve for the Fibo-SPMAC and ECP-SPMAC is NIST(p192). The key and nonce for Fibo-SPMAC and ECP-SPMAC are 160 bit. Since ECDLP is harder to solve than traditional discrete logarithm problems, the length of the key in the elliptic curve cyclic group-based algorithm is shorter than that based on the multiplicative cyclic group [8].

The code of both the client and verifier is implemented in Python 3.7. For the first experiment, the code is set up for several IoT devices like Android and Raspberry Pi. The other three experiments are set up on two Raspberry Pis with 1GB RAM and 1 CPU core, which is widely used in IoT simulation [39][58].

3.4.1 Performance of Four Schemes under Different Devices

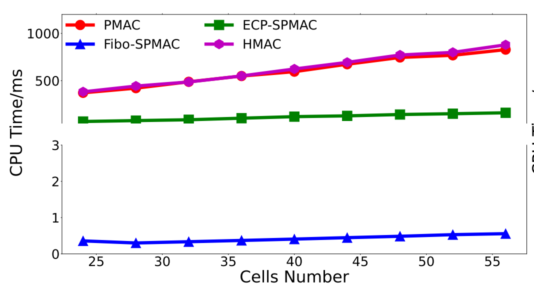
In this subsection, we evaluate the performance of Fibo-SPMAC, ECP-SPMAC, PMAC, and HMAC in different IoT devices, which contain Android and Raspberry Pi. The number of cells is 60, and the number of disclosed cells is 20. Table 3.4 summarizes the performance. The results in red mean the best performance among the

Table 3.4: Performance of HMAC, PMAC, and SPMAC in Different Devices

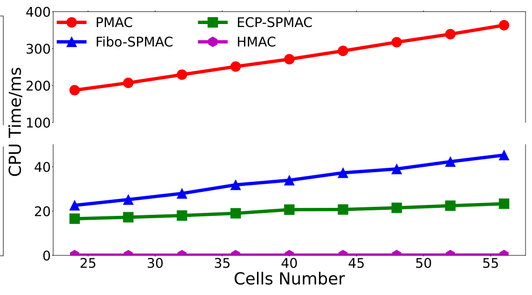
Device	Performance	**PMAC**	**Fibo-SPMAC**	**ECP-SPMAC**	**HMAC**
	$T_{MG}(ms)$	2722.351	10.552	70.790	2633.401
Android 1	$T_V(ms)$	906.236	5.770	24.293	882.716
	MAC size(byte)	82	32	32	3865
	$T_{MG}(ms)$	2793.060	10.532	68.210	2716.082
Android 2	$T_V(ms)$	943.201	5.493	24.671	932.321
	MAC size(byte)	82	32	32	3865
	$T_{MG}(ms)$	1333.684	48.062	188.269	987.822
Raspberry Pi	$T_V(ms)$	433.207	25.832	70.173	303.230
	MAC size(byte)	82	32	32	3865

three IoT devices. We compare the time for mapping and MAC generation (T_{MG}), the time for MAC verification (T_V), and the MAC size. The RAM in Android 1 is 2GB, and the number of CPU cores is 1. The RAM in Android 2 is 512 MB, and the number of CPU cores is 1. The RAM in Raspberry Pi 1 is 1GB, and the number of CPU cores is 1.

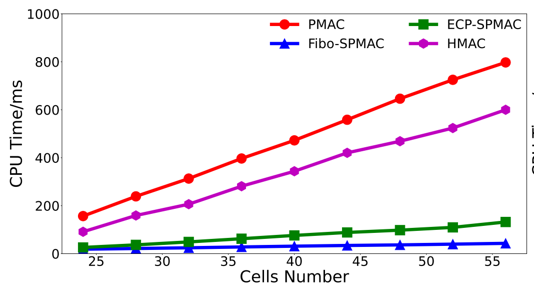
Generally speaking, all three schemes can be applied to different IoT devices. The HMAC has a significantly larger size MAC than the others, while the SPMAC is the smallest one. In other words, Our proposed scheme has a better performance in the IoT scenario. In PMAC and HMAC schemes, the performance of Android is much worse than that in Raspberry Pi. When comparing the performance of Fibo-SPMAC and ECP-SPMAC, ECP-SPMAC consumes more time on T_{MG} and T_V. Therefore, Fibo-SPMAC performs better than ECP-SPMAC in general. The resources in IoT devices are limited, so we use Raspberry Pi as the IoT device in the rest of the experiments.



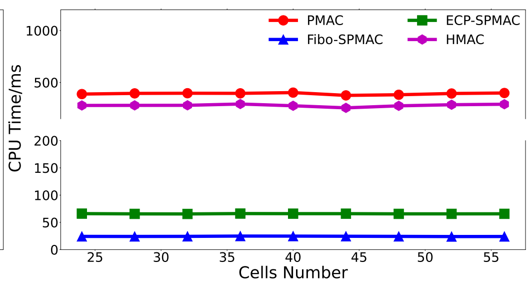
(a) Time for Mapping



(b) Time for MAC Generation



(c) Time for Suffix Construction

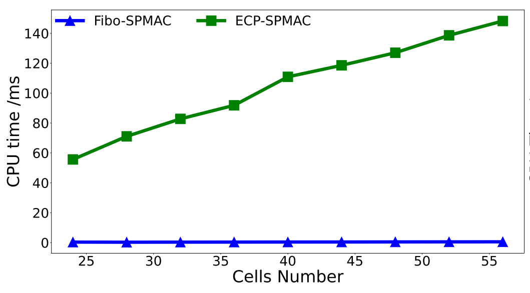


(d) Time for MAC Verification

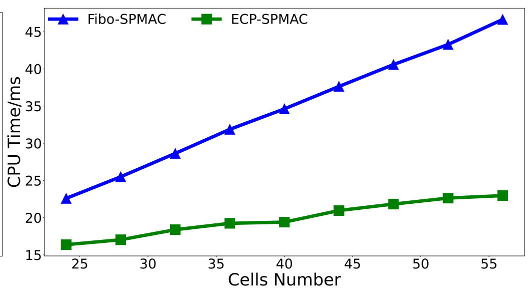
Figure 3.5: Time Consumption among SPMAC, PMAC, and HMAC in Prefix-Suffix Model

3.4.2 Performance of Four Schemes under Different Number of Cells

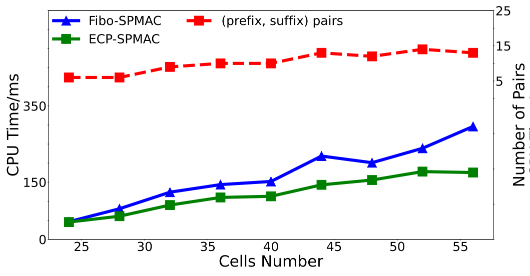
In the following experiments, we compare the performance of PMAC, Fibo-SPMAC, ECP-SPMAC, and HMAC on one message with different numbers of cells. In this experiment, the number of cells changes from 24 to 60, and the number of disclosed cells is 18. According to Figure 3.5(a), we observe that the prime mapping consumes the longest time among the four schemes, and the Fibo mapping is the least one. Computing the quadratic residue is much more complicated than finding a group of Fibonacci numbers. Consequently, the time for ECP mapping is long. Moreover, since the computation resource in IoT devices is limited, mapping each cell to a distinct prime consumes the largest amount of time. Thus, the time for mapping in PMAC and HMAC is long. The suffix construction and MAC verification contain mapping.



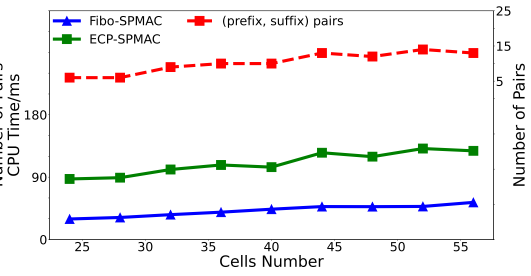
(a) Time for Mapping



(b) Time for MAC Generation



(c) Time for Suffix Construction



(d) Time for MAC Verification

Figure 3.6: Time Consumption between Fibo-SPMAC and ECP-SPMAC in Random Model

Thus, the performance of HMAC and PMAC is worse than that of SPMAC.

According to Figure 3.6(a), the time for Fibo mapping is shorter than that in ECP mapping. The reason has been explained in Chapter 3.4.2. The time for MAC generation in Fibo-SPMAC consumes more time than that in ECP-SPMAC. Recall that Eq. 3.2 consists of scalar multiplication and addition, while Eq. 3.6 only contains addition operation. Consequently, the time consumption of Fibo-SPMAC generation is enormous.

According to Figure 3.6(c), the time for suffix construction increases when the number of cells increases. With the stable number of disclosed cells, the number of cells increasing leads to the growth of the number of useless cells. Therefore, the client should spend more time on suffix construction. Since the number of disclosed cells is stable, the time consumption of MAC verification does not change in Figure 3.6(d).

In summary, according to Chapter 3.4.1, we can conclude that SPMAC has a **better performance** in IoT scenarios than PMAC and HMAC, especially in efficiency.

3.4.3 Performance of Four Schemes under Different Number of Disclosed Cells

In this experiment, the number of disclosed cells changes from 12 to 56, and the number of cells is 60.

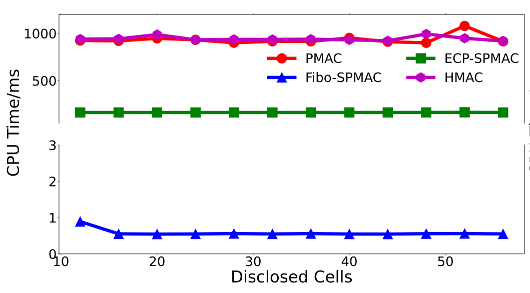
According to Figure 3.7(a) and Figure 3.7(b), we observe that the time for mapping and MAC generation in all schemes do not change since both of them are based on the number of cells. The time of prime mapping is about 5.68 times larger than that in ECP-SPMAC, 1611.65 times larger than that in Fibo-SPMAC, and nearly the same as that in HMAC.

In Figure 3.7(c) and Figure 3.7(d), with the increasing number of disclosed cells, the time for suffix construction decreases, and that for MAC verification increases. Moreover, the decrease rate of PMAC is faster than that of SPMAC. The time for mapping occupies most of the time in the PMAC scheme. When the number of disclosed cells increases, the number of useless cells decreases. Thus, the time needed to map the useless cells drops dramatically in PMAC schemes. On the contrary, the verifier spends more time on mapping.

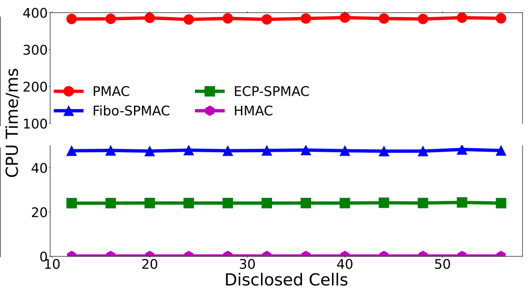
Since the number of cells does not change in this scheme, the time for mapping and time for MAC generation does not change in Figure 3.8(a) and Figure 3.8(b).

The time for MAC generation in Fibo-SPMAC is larger than that in ECP-PMAC, which has been explained in Chapter 3.4.2. As shown in Figure 3.8(c) and 3.8(d), when the number of disclosed cells increases, the time for MAC verification grows up, and the time for suffix construction declines.

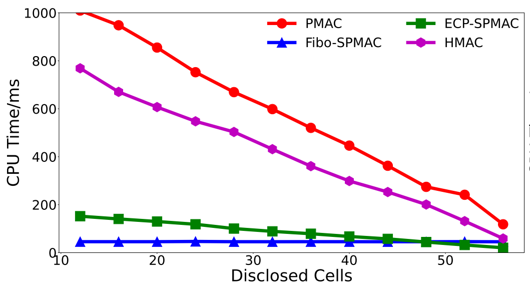
Moreover, the time for suffix construction remains relatively high. A similar situation



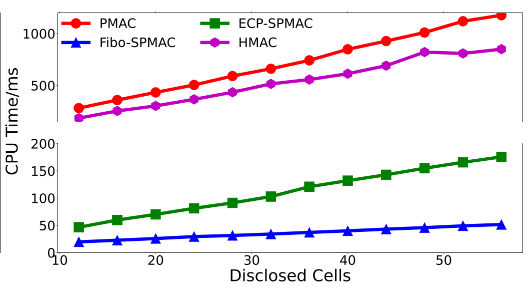
(a) Time for Mapping



(b) Time for MAC Size



(c) Time for Suffix Construction



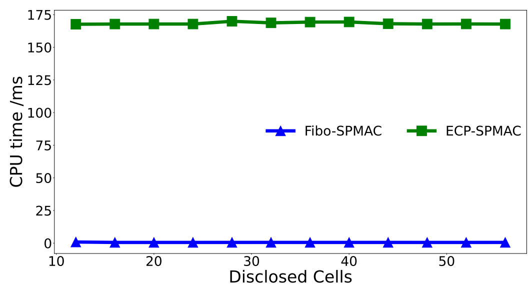
(d) Time for MAC Verification

Figure 3.7: Time Consumption among SPMAC, PMAC, and HMAC in Prefix-Suffix Model

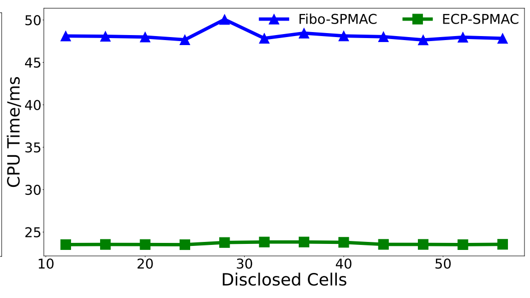
happens in Figure 3.6(c). The reason is that the disclosed cells are randomly chosen, which changes the number of $(\text{prefix}, \text{suffix})$ pairs. Once the number of $(\text{prefix}, \text{suffix})$ pairs changes sharply, it reflects on the time for suffix construction. Recall the Eq. 3.3 and 3.7, the larger the number of $(\text{prefix}, \text{suffix})$ pairs is, the more number of suffixes needed to be constructed. Thus, the client should spend more time on computing σ_i. In the meantime, according to Eq. 3.5 and 3.8, the time for combining σ_i and $pre_i(\mathbf{x})$ changes in Figure 3.8(d).

3.4.4 Practical Application

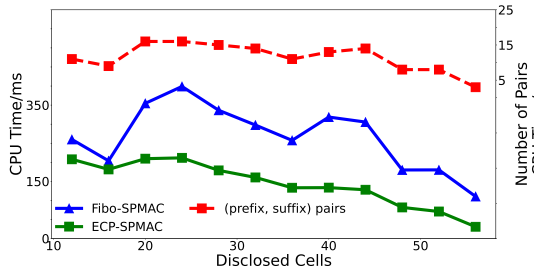
This experiment compares Fibo-SPMAC and ECP-SPMAC in a practical situation. One thousand messages need to be transmitted, and 50% of the cells in the message are disclosed. The experiment compares the time for suffix construction (T_C), time



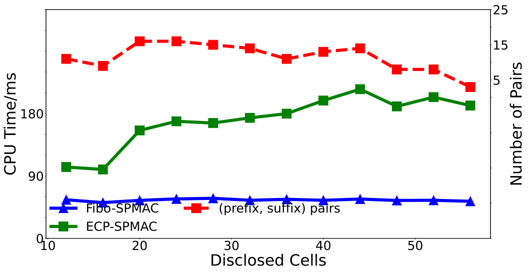
(a) Time for Mapping



(b) Time for MAC Generation



(c) Time for Suffix Construction



(d) Time for MAC Verification

Figure 3.8: Time Consumption between Fibo-SPMAC and ECP-SPMAC in Random Model

for MAC verification (T_V), and extra communication costs. The extra communication cost is computed according to the MAC size and the mapping look-up table.

Recall that ECP-mapping consumes more time than Fibo-mapping according to Figure 3.7 because ECP-mapping is a probabilistic algorithm. If the client completes the mapping function and shares the mapping look-up table with the verifier, they can finish the mapping function quickly. The client should transmit a mapping look-up table through a secure channel.

According to Figure 3.9, we can find out that both T_C and T_V in Fibo-SPMAC do not change. On the opposite side, T_C and T_V decrease a lot in ECP-SPMAC. The reason is that the client or verifier can check the mapping look-up table directly instead of using the mapping function to compute. The mapping process saves time when computing quadratic residue. However, Fibo-mapping does not consume a lot of time. Thus, the time consumption does not decrease significantly.

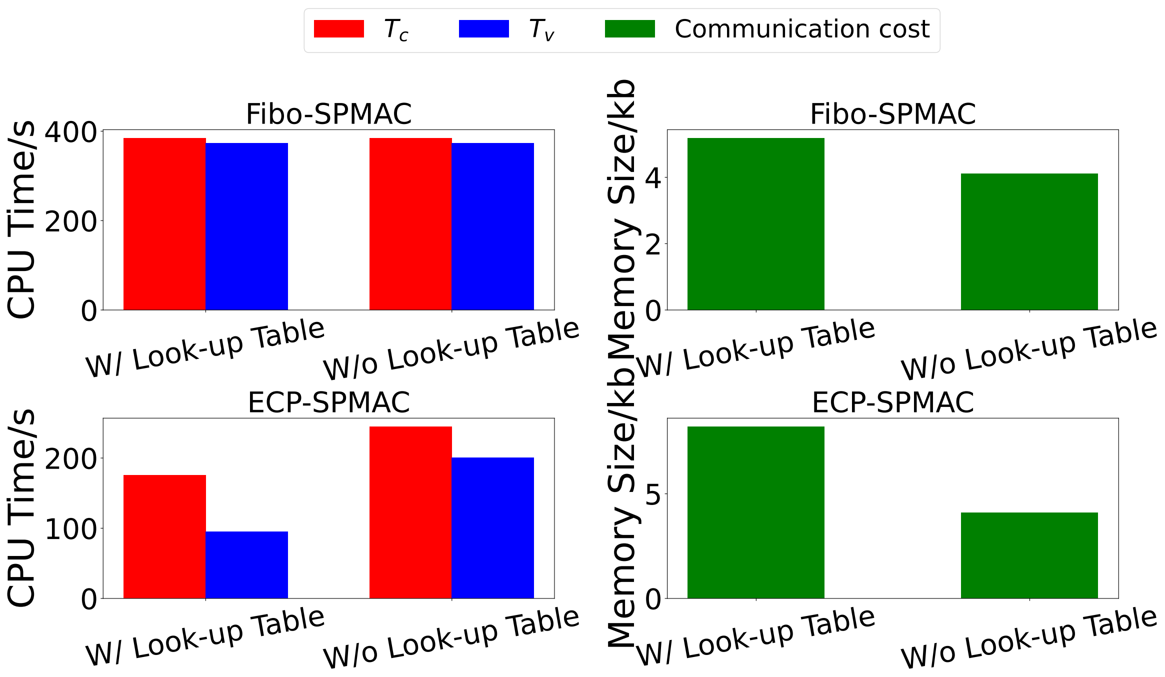


Figure 3.9: Time Consumption and Communication Cost on Fibo-SPMAC and ECP-SPMAC with or without Mapping Look-up Table

Generating a mapping table consumes more time and memory, but the client only needs to do this one time, which can save time for suffix construction and MAC verification. Thus, it can be advantageous for the client as well as the verifier in the future. Furthermore, if we apply a mapping look-up table in SPMAC schemes, a trusted third party can help generate it for further use. Using a mapping look-up table instead of the mapping function increases the communication cost in Fibo-SPMAC and ECP-SPMAC since the mapping look-up table should be shared between the client and the verifier.

3.5 Summary

In this chapter, we demonstrate the subset privacy problem and propose two new schemes: Scalable Prefix Verifiable Message Authentication Code on Fibonacci Mapping (Fibo-SPMAC) and Scalable Prefix Verifiable Message Authentication Code on Elliptic Curve Point Mapping (ECP-SPMAC). Both schemes can verify the message integrity of any disclosed cells in the message and inherit the advantages of PMAC.

Furthermore, the experimental results show that the proposed schemes are suitable for IoT scenarios and have a better efficiency performance. According to Chapter 3.4.2 and 3.4.3, the time for mapping in ECP-SPMAC is larger than the time for suffix construction and PMAC verification. Moreover, Figure 3.9 shows that when we apply a look-up table in ECP-SPMAC to decrease the time for mapping, the time for suffix construction and SPMAC verification decreases. On the contrary, the look-up table has a limited effect on the time consumption of Fibo-SPMAC. It is fine to spend more time creating a mapping table, which is a one-time SPMAC generation that the client needs to complete. Subsequently, the client and verifier can complete suffix construction and MAC verification faster. Consequently, ECP-SPMAC performs better when the client and verifier share a large bandwidth channel or the communication cost is not essential. On the contrary, Fibo-SPMAC is suitable when the mapping function should be done by both the client and the verifier, which means the channel bandwidth is limited. In Chapter 3.4.4, the experiment result proves our assumption.

Chapter 4

Subset Integrity between Two Entities

In this chapter, we begin to study the subset integrity problem between two entities. With IoT providing great convenience, message integrity in IoT becomes more critical. In traditional message verification schemes, the verifier is supposed to have an infinite workload and energy to verify all the incoming messages, while IoT devices have computation ability and energy limitations in a practical situation. Therefore, they have a bottleneck in verifying the integrity of all the incoming messages, especially when the messages are large in volume and high in frequency. In this situation, a secure fog architecture [85] is proposed to facilitate data transmission. For instance, in the manufacturing industry, IoT devices can monitor the machine's operational status and periodically transmit the message to the manager. However, the manager does not have enough ability to verify all the incoming messages. Hence, a fog node with great computation ability can assist in verifying a portion of the message. Nevertheless, the computation ability of the fog node cannot be exhausted for verification because it should undertake some additional tasks (e.g., data analysis or data training).

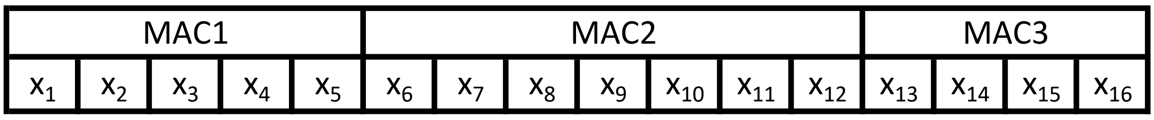
Suppose the message contains several cells (e.g., x_i in Figure 4.1(a)). The sender generates MAC_1 for the whole message and MAC_2 for a subset of the message (e.g., x_1 to x_7). Specifically, we call it static verification due to the fog node can verify the cells, which the sender determines. However, the static solution is not viable in practical situations. The sender does not have prior knowledge of the computation ability in the fog node since the fog node and the data sources belong to different owners. Insufficient utilization of the fog node wastes the computation ability. Conversely, excessive utilization of the fog node surpasses its computation ability. Another naive solution is to generate and concatenate each cell's MAC (e.g., key-hashed MAC [55]). However, the size of the final MAC is proportional to the number of cells, which leads to great communication overhead.

To address the problem, we consider whether the fog node can randomly verify a subset of messages like Figure 4.1(b) according to its computation ability. The fog node can choose random cells to verify (e.g., x_1 to x_5 or x_9 to x_{14}). Then it computes $F(\text{MAC})$. The integrity of them can be guaranteed if $F(\text{MAC})$ is related to MAC (e.g., $F(\text{MAC}) \subseteq \text{MAC}$) so that the fog node can verify as many cells as it can. F is the transformation function. For instance, when the fog node selects a subset of the message, if the number of cells reaches the determined threshold, the regenerated MAC is the same as the MAC of the whole message.

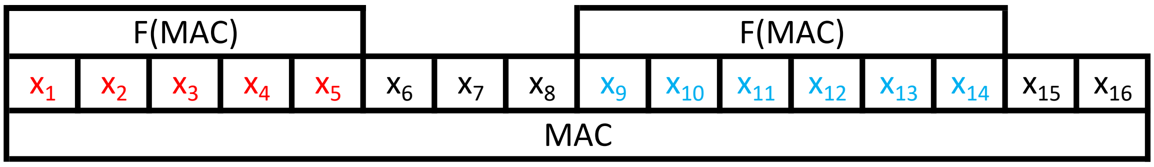
In this chapter, we design a novel MAC scheme called "Partial Verification Message Authentication Code" (PV-MAC) for the IIoT scenario. In this scheme, the fog node can randomly and optimally choose the cells it can verify according to its computation ability. Consequently, the message receiver is only required to verify the cells that remain unverified by the fog node.

In summary, the contributions we make in this chapter are as follows:

1. We propose PV-MAC for subset verification in a fog-based IIoT scenario. The MAC is generated based on the whole message, and the fog node can randomly choose cells for verification rather than being determined statically by the mes-


MAC1
MAC2
MAC3
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15} x_{16}

(a) Static Verification


F(MAC)
F(MAC)
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15} x_{16}
MAC

(b) Subset Verification

Figure 4.1: Static and Subset Verification

sage sender.

2. We propose PV-CMAC to decrease the communication cost in the IIoT scenario.

3. We propose an optimized solution to minimize the energy consumption for the message receiver.

4. We conduct a theoretical and empirical analysis to evaluate the performance of PV-MAC.

The rest of the chapter is organized as follows. Chapter 4.1 defines the problems and the network model we discuss. Chapter 4.2 introduces the syntax of PV-MAC, and Chapter 4.4 presents the energy optimization problem in PV-MAC. Chapter 4.3 shows the theoretical analysis of performance and security. In Chapter 4.6, the experimental results show the performance of the proposed schemes. Chapter 4.7 summarizes this chapter.

4.1 Problem Definition and System Model

4.1.1 System Model

Figure 4.2 shows the system model of subset verification. There are three entities: data sources (DS), a fog node (FN), and data users (DU). The data sources own the messages and send them to the data user. The data users are IoT devices. They require the message from the DS. However, they do not have enough computation ability to confirm the accuracy of every cell in the incoming messages. The FN [3] has strong computation ability and can be applied to offload the heavy verification on the data user. Moreover, the FN can provide smaller service delays than cloud computing so that one FN can provide compromising service in the system. The FN randomly chooses the cells (e.g., the vertically shaded cells) to verify. Then, the DU verifies the remaining cells (e.g., the horizontally shaded cells). In the IIoT scenario, a reliable FN can be applied, which means the FN is trusted. In the meantime, the communication channel between FN and DU is dependable [85] so that the DU can trust the verification result from the FN. We suppose the FN verifies a subset of cells in the incoming message, and the DU only needs to validate the rest of the cells.

The energy consumption is relevant to the number of verified cells, which has two properties:

1. The computation energy consumption always increases as the computation amount increases.
2. The marginal energy consumption for the fog node is increasing.

In a mathematical description, energy consumption and its growth rate are increasing monopoly. In this chapter, the energy model follows like Eq. 4.1 [27][62]. a, b, and e are energy parameters larger than 0. τ is the computation amount, which can be considered as the number of verified cells.

$$E(\tau) = a\tau^2 + b\tau + e \tag{4.1}$$

In the meantime, we apply the workload to represent the computation ability, which means the number of cells that the FN can verify. We use the workload to measure the computation ability of the fog node. In other words, it is relevant to the number and size of the cells. The workload model follows like Eq. 4.2. B is the size of cells, which has been predetermined. The workload of the FN cannot exceed its computation ability.

$$W(\tau) = B\tau \tag{4.2}$$

Besides the system model, we define the adversary model. As security, a MAC scheme should satisfy unforgeability. The adversary follows the Dolev–Yao (DY) Adversary model [65], which has been widely used in formal analysis of security protocols. Specifically, when DS delivers the messages and MACs to the FN, a Probabilistic Polynomial Time (PPT) Attacker [35] attempts to falsify the messages by executing a polynomial time algorithm while leaving the MACs unchanged. Since the FN is reliable and owned by the DU, the PPT attacker can only modify the messages during the transmission from DS to FN.

4.1.2 Problem Definition

In this subsection, we discuss how to achieve subset integrity verification in a Fog-based IIoT scenario. Specifically, the fog node can **randomly** and **optimally** choose cells to verify before the DU according to its computation ability and energy in IIoT scenarios. Therefore, this problem is split into two parts: subset verification and energy optimization problem.

Formally, we consider that the message contains several cells (e.g., $\mathbf{x} = (x_1, x_2, \ldots, x_n)$). The definition of subset verification is as follows,

Definition 3. ***(Subset Verification)****: The fog node can* ***randomly*** *choose a subset in one message for verification* ***by himself****.*

$$\forall \boldsymbol{x}_v \subseteq \boldsymbol{x}, \quad f(\boldsymbol{x}) = \mathcal{F}(f(\boldsymbol{x}_v)) \tag{4.3}$$

In Eq. 4.3, $\mathbf{x}$ is the original message, and $\mathbf{x}_v$ is the subset of $\mathbf{x}$, which are verified by the FN. f is the MAC generation function for the whole message. $\mathcal{F}$ is a translation from the original MAC to the MAC of $\mathbf{x}_v$.

Based on the subset verification, the optimization problem [25] is essential. The IoT devices have limited workloads and energy. Therefore, the FN should undertake as many verification tasks as possible since it has a stronger computation ability than IoT devices. However, the FN still confronts some limitations due to the hardware (e.g., upper-bound workload). In summary, the optimization problem follows like Eq. 4.4.

$$\begin{aligned} \min \quad & E_{DU} \\ \textbf{s.t.} \quad & W(\mathbf{x}_v) \le W_{FN}^{\max} \end{aligned} \tag{4.4}$$

Specifically, $W_{FN}^{\max}$ is the upper-bound workload of the FN, and E_{DU} is the energy consumption and the objective function, which is relevant to the number of verified cells. The details will be given in Chapter 4.4.

4.2 Partial Verification Message Authentication Code

In this section, we introduce the solution for the subset verification problem in the IoT scenario. In this situation, the clients outsource the verification tasks to a trusted fog node. Formally, the Partial Verification Message Authentication Code (PV-MAC) for the fog-based IIoT scenario. PV-MAC consists of Hierarchical Secret Sharing Based Message Authentication Code (HSS-MAC) and SPMAC [103]. The two MACs are generated based on the whole message and are necessary to solve the subset verification in practical situations.

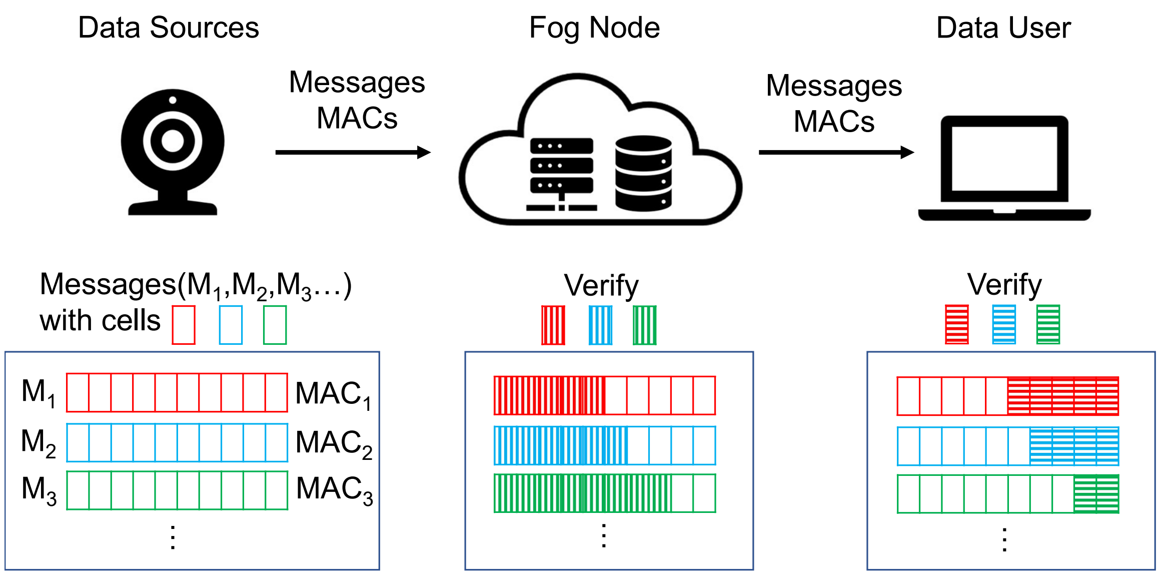


Figure 4.2: System Model of Subset Verification

4.2.1 Secret Sharing based Message Authentication Code

Generally, to achieve dynamic verification, we can combine secret sharing and MAC together, and propose secret sharing based MAC (SS-MAC) [73]. Algorithm 5 shows how it works.

Insides, PRF is a pseudorandom function, and SS_r is the secret recovery function. In summary, the FN can select at least t cells to recover the secret. Then, he can regenerate the MAC according to the secret, symmetric key, and the PRF. We use a concrete example to demonstrate how it works.

Example 4. *Suppose the message is* $(1, 2, 3, 4, 5)$, *the secret(s) to be shared is 13, and the prime for secret sharing is 17. The polynomial of the secret sharing is* $f(x) = 13 + 10x + 2x^2$. *Then, the shadows can be computed as*

$$(1, 8), (2, 7), (3, 10), (4, 0), (5, 11)$$

Afterward, the DS can generate the SS-MAC according to secret, $k_{SS} = 2$, *and chosen PRF according to Eq. 4.5. Then he sends the processed message* $(1, 2, 3, 4, 5, 8, 7, 10, 0, 11)$

Algorithm 5 Framework of SS-MAC

```
Input: Symmetric key for SS-MAC $k_{\mathrm{SS}}$; message $\mathbf{x}$; the polynomial of the secret
    sharing $f()$
Output: SS-MAC
 1: DS, FN receives the symmetric key $k_{\mathrm{SS}}$
 2: DS $\leftarrow s$,
 3: for $i = 1; i < N : i++$ do
 4:     DS $\leftarrow f(\mathsf{x_i})$
 5: end for
 6: SS-MAC $= PRF(k_{\mathrm{SS}}, s)$
 7: FN $\leftarrow \mathbf{x}$, $f(\mathbf{x})$, SS-MAC
Input: Symmetric key for SS-MAC $k_{\mathrm{SS}}$; verified subset message $\mathbf{x}_v$; corresponding
    subshares $f(\mathbf{x}_v)$
Output: 1 or 0
 8: FN recovers $s$ according to the cells he wants to verify $(x_v)$ and their corresponding
    subshares $(f(\mathbf{x}_v))$.
 9: $s' \leftarrow SS_r(x_v, f(\mathbf{x}_v))$
10: SS-MAC' $= PRF(k_{\mathrm{SS}}, s')$
11: if SS-MAC == SS-MAC' then
12:     return 1
13: else
14:     return 0
15: end if
```

and SS-MAC to the FN.

$$\text{SS-MAC} = PRF(k_{SS}, s) \tag{4.5}$$

Upon receiving the message, the FN can select three or more shadows(e.g., (1, 8), (2, 7), (3, 10)) and recover the secret. Then he can regenerate the SS-MAC and check whether it is equal to the received one.

However, if we only applied SS-MAC in the subset verification problem. it confronts a new problem call **substitution attack**. The adversary can replace one of the shadows without changing the secret. Recall the Example 4, the fog node recovers the secret even if one of the shadows is replaced by $(6, 9)$. Therefore, the output of the fog node will be either negative or false positive, which makes the feedback meaningless. Consequently, we need to find other methods to fulfill the request.

4.2.2 Hierarchical Secret Sharing Based Message Authentication Code

In this part, we introduce the syntax of HSS-MAC. Similar to the SS-MAC, HSS-MAC is based on hierarchical secret sharing. The notations are in Table 4.1.

HSS-MAC can address the substitution attack. Compared with SS-MAC, HSS-MAC separates the message into different levels. Moreover, the levels of each cell remain constant and can be privately shared with the FN before transmission. Accordingly, the attacker cannot recover the secret to a high degree, which will be proved in Chapter 4.3.2. In addition, the SPMAC can further avoid the Substitution Attack. Algorithm 6 shows how HSS-MAC works.

Algorithm 6 Framework of HSS-MAC

```
Input: Symmetric key $\beta$; message $\mathbf{x}$;
Output: HSS-MAC
 1: DS, FN DS, FN receives the symmetric key $\beta$
 2: DS receives $s$ and all the $f_i()$,
 3: for $i = 1; i < L : i{+}{+}$ do
 4:     for $j = 1; j < N : j{+}{+}$ do
 5:         DS computes $f_i(\mathsf{x_j})$
 6:     end for
 7: end for
 8: HSS-MAC = $PRF(\beta, s)$
 9: FN $\leftarrow \mathbf{x}$, $f(\mathbf{x})$
Input: Symmetric key $\beta$ message $\mathbf{x}$; subshares $f(\mathbf{x})$
Output: 1 or 0
10: FN chooses random cells in the message, which achieve secret recovery in HSS,
    denoted by $x_{\mathrm{v}}$.
11: FN recovers the secret $s'$ according to HSS.
12: FN computes HSS-MAC' = $PRF(\beta, s')$.
13: if HSS-MAC' == HSS-MAC then
14:     return 1
15: else
16:     return 0
17: end if
```

Table 4.1: Variables List for HSS-MAC and PV-MAC

Parameter	**Instruction**
s	The master secret
$\mathbf{x}$	The original message
$\mathbf{x}_v$, $\mathbf{x}_r$	The cells verified by the FN and DU
$f_i()$	The polynomial function of in level i
$f(\mathbf{x})$	The collection of the subshares
PRF	The Pseudorandom Function for MAC generation
L	The number of levels in one message
N	The number of cells in one message
α, β	The symmetric key for PV-MAC
HSS_r	The master secret recovery function
SS_r	The secret recovery function is secret sharing
t_i	The threshold in level i

4.2.3 The Syntax of PV-MAC

In this subsection, we introduce the syntax of PV-MAC. The generic PV-MAC scheme consists of four phases: System Initialization Phase, MAC Generation Phase, Fog Node Verification Phase, and Data User Verification Phase. Figure 4.3 shows the process of MAC generation and verification.

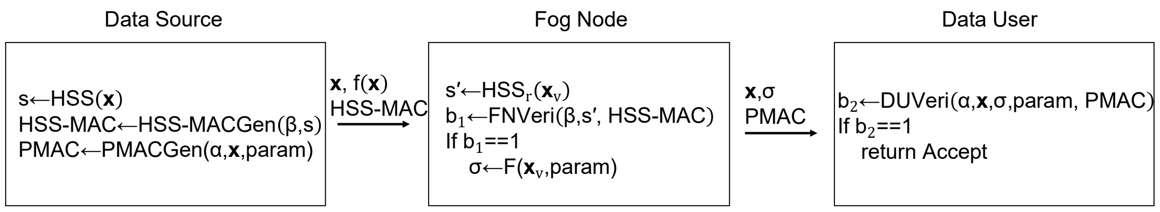


Figure 4.3: PV-MAC Generation and Verification Phase

In the System Initialization Phase, the DS, FN, and DU initialize the system by the following steps,

1 Based on the security parameter λ, the DS gets the two symmetric MAC keys α and β.

2 After that, the DU receives α, and FN receives β and the number of level L.

3 The DU selects a large prime number p and a generator g for a multiplicative group. These parameters are publicly known.

In the MAC Generation Phase, when the DS sends the message to the DU, it generates two MACs: HSS-MAC and SPMAC for the whole message.

1 DS generates a master secret s for the HSS scheme, separates the cells into different levels, and generates the subshares for them. Specifically, the DS applies a polynomial function $f_i()$ for ith level. Then the subshare of jth cell (x_j) is computed as $f_i(x_j)$.

2 Subsequently, the HSS-MAC is generated according to Eq. 4.6.

$$\text{HSS-MAC} = PRF(\beta, s) \tag{4.6}$$

3 Subsequently, the DS selects a random nonce $\gamma < p$ and generates SPMAC according to $\mathbf{x}$, α, γ and the public parameters.

$$\text{SPMAC} = \alpha(\Pi(\mathbf{x}) + \gamma)g \bmod p \tag{4.7}$$

In the Fog Node Verification Phase, since the DU does not have enough computation ability to verify the message, the FN verifies a subset of the message according to HSS-MAC. Since the FN has greater computation ability than DU, the number of verified cells can be larger than the threshold, which satisfies the HSS recovery.

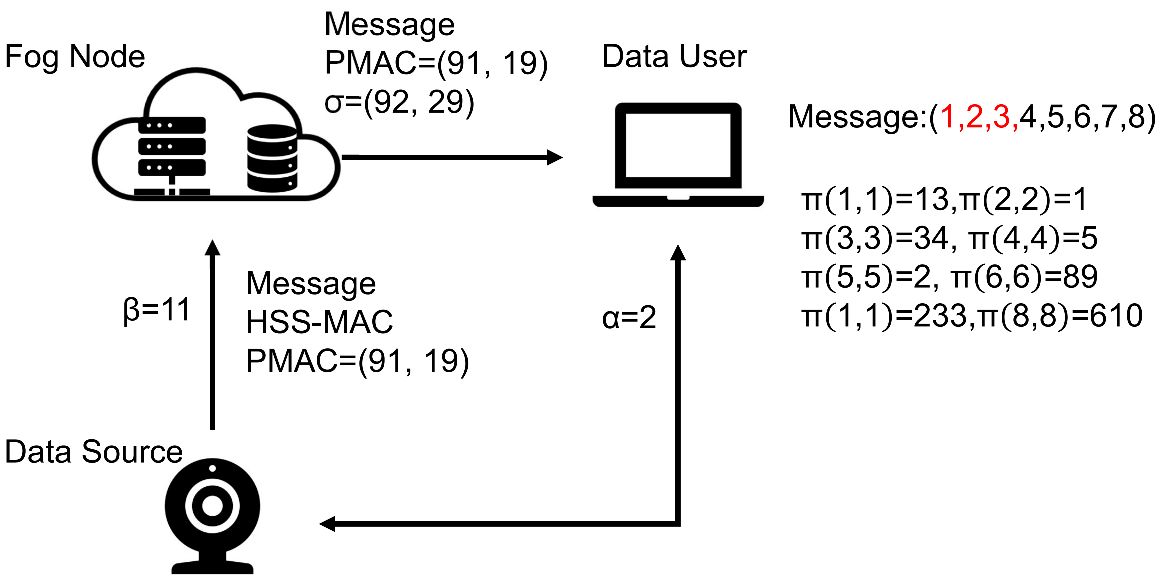


Figure 4.4: Concrete Example of PV-MAC

1 The FN chooses the cells ($\mathbf{x}_v$) from different levels and recover a secret (s').

$$HSS_r(\mathbf{x}_v) = s' \tag{4.8}$$

2 Subsequently, the FN verifies $\mathbf{x}_v$ based on s' and β. If $\mathbf{x}_v$ has not been modified, it computes σ according to Eq. 4.9 and sends it with the message to the DU.

$$\sigma = (\Pi(\mathbf{x}_v) + \gamma)g \bmod p \tag{4.9}$$

In the Data User Verification Phase, once the DU receives the verification result from FN, it can verify the rest of the cells ($\mathbf{x}_r$) in the message.

According to $\mathbf{x}_r$, α, γ and the public parameters, the DU can reconstruct the SPMAC according to Eq. 4.10.

$$\text{SPMAC}' = \alpha\,(\sigma + \Pi(\mathbf{x}_r)g) \bmod p \tag{4.10}$$

We exemplify how PV-MAC works according to Figure 4.4 and Example 5.

Example 5. *Suppose the message is (1,2,3,4,5,6,7,8), and divided into two levels. (1,2,3,4) belongs to the first level, and (5,6,7,8) belongs to the second level. The* *thresholds of each level are 3 and 3. The prime for HSS is 97, and the master secret is 50.*

First, the DS generates the polynomials of each level.

$$f_1(x) = 50 + 38x + 42x^2 \bmod 97$$
$$f_2(x) = 29 + 16x + 65x^2 \bmod 97$$

Then it computes the subshares according to the cells, and the processed message follows,

$$(1, 2, 3, 4, 5, 6, 7, 8, 33, 3, 57, 1, 85, 40, 28, 49)$$

The first eight cells are the original message, while the rest are the subshares. According to the master secret and the symmetric key $\beta = 11$*, the DS can compute the HSS-MAC by PRF. Then DS generates the SPMAC according to* $\alpha = 2$*,* $g = (72, 29)$*,* $p = 97$*, and Eq. 4.7. The result is* $\text{SPMAC} = (91, 19)$*.*

The FN can recover the master secret by selecting the first three cells (e.g., $(1, 2, 3)$ *in red), and the HSS-MAC is then recomputed. Afterward, the FN computes the suffix (*$\sigma = (92, 29)$*) according to Eq. 4.9 and sends the original message and suffix to the DU.*

After receiving the suffix, the DU can verify the SPMAC according to Eq. 4.10 and check whether the remaining cells are modified.

In summary, the two components in PV-MAC are generated based on the whole message. They have different purposes and are complementary to each other. HSS-MAC provides subset verification for the FN. However, if we only apply HSS-MAC in this network, the DU cannot verify the rest of the cells. According to Chapter 2.2.2, most of the cells in the highest level have been verified by the FN since it has stronger computation ability. Without enough number of them, the DU cannot reconstruct the

HSS-MAC based on $\mathbf{x}_r$. In this situation, the DU verifies SPMAC without verifying $\mathbf{x}_v$ one more time. Simultaneously, if we only apply SPMAC in this network, the FN only conducts the computation tasks and cannot find the unauthorized modification on $\mathbf{x}_v$. Therefore, the two components are essential for this scheme. Furthermore, there is still one constraint for the FN, which is the number of verified cells should be larger than the threshold of HSS-MAC. Since the FN has a larger computation ability than the DU, he can select more cells for verification. In general, the number of verified cells is larger than the threshold.

4.3 Theoretical Analysis

In this section, we theoretically analyze the time consumption and security of the PV-MAC.

4.3.1 Time Consumption of SS-MAC and HSS-MAC

The time consumption of secret sharing is Eq. 4.11[6] when FN chooses t cells.

$$T_{\mathrm{SS}_r} = O(t \log^2(t)) \tag{4.11}$$

Similarly, the time consumption of HSS recovery is as follows,

$$T_{\mathrm{HSS}_r} = O(\sum_{i=1}^{l} t_i \log^2(t_i)) \tag{4.12}$$

Based on Eq. 4.11 and Eq. 4.12, we can deduce Theorem 6.

Theorem 6. *If the FN chooses the same number of cells for verifying, the time consumption of HSS-MAC is smaller than that of SS-MAC.*

Proof. When the FN verifies SS-MAC and HSS-MAC, it should recover the secret and compute the PRF. Therefore, the difference between the two schemes is the time consumption of secret recovery. The number of cells verified by the FN in SS-MAC is the same as that in HSS-MAC. Consequently, we can derive that,

$$T_{\mathrm{SS}_r} = O((\sum_{i=1}^{l} t_i) \log^2(t))$$

Since $tlog^2(t)$ is monotonically increasing and $\forall i \quad t_i \leq t$, we can deduct that

$$\forall i \in l, O(t_i \log^2(t_i)) \leq O(t_i \log^2(t))$$

We can prove that $T_{\mathrm{HSS}_r} < T_{\mathrm{SS}_r}$ □

4.3.2 Security of PV-MAC

PV-MAC consists of the SPMAC and the HSS-MAC. The security of SPMAC has been introduced in Chapter 3.3. Specifically, HSS-MAC combines PRF and HSS. In the HSS scheme, the subshares from one level cannot be used for another level. Consequently, we can derive Theorem 7.

Theorem 7. *The probability of substitution attack in HSS is* $(\frac{C_{n_1}^{t_1}}{C_n^{t_1}})$.

Proof. Suppose an attacker can recover the secret in HSS without knowing the level. In that case, the attacker needs at least t_1 shadows from the N shadows to recover the secret and achieve the substitution attack.

Since the number of cells is N and the number of cells in the highest level is n_1, the probability of the attack is $(\frac{C_{N_1}^{t_1}}{C_N^{t_1}})$. □

In addition, we should prove the unforgeability, which is defined as follows,

Definition 4. ***(Unforgeability)***. *HSS-MAC (Setup, Sign, Verify) is unforgeable under adaptive chosen attacks if for any efficient algorithm $\mathcal{A}$ that the experiment Unforgeability$_{\mathcal{A}}^{\text{HSS-MAC}}$ evaluates to 1 is negligible.*

Experiment Unforgeability$_{\mathcal{A}}^{\text{HSS-MAC}}$
-$\beta \leftarrow Setup(1^{\lambda})$
-$HSS\text{-}MAC \leftarrow \mathcal{A}^{Gen(\beta,\dot{)}}(\beta\ )$
-*for* $i = 1, 2, ..., q$ *denoted by HSS-MAC*$_i$,
the queries to the oracle($\mathcal{O}$) *Sign return 1*
iff Verify$(\beta, HSS - MAC) = 1$

Specifically, $\mathcal{A}$ generates a random key from $Setup(1^{\lambda})$. Then $\mathcal{A}$ submits MAC queries by sending different messages ($\mathbf{x}_i$) and receives HSS-MAC$_i$ from the Oracle ($\mathcal{O}$). $\mathcal{A}$ wins if it can modify the messages by executing a polynomial time algorithm while leaving the MACs unchanged.

Theorem 8. *Consider the HSS-MAC is generated according to a $(q, T, \epsilon)-$ PRF family, and the length of MAC is L. The HSS-MAC is secure with respect to Definition. 4. In other words, for any attacker $\mathcal{A}$ who runs in time at most T, makes at most q queries, the probabilities of success is upper-bounded by*

$$\Pr[\mathcal{A}\ wins] \leq \epsilon + \tfrac{Q^2}{2^L}$$

Proof. Assume that there is an adversary $\mathcal{A}$ that breaks the HSS-MACGen security. Then, it can also break the security of PRF families. A sequence of games $G_0 - G_3$ will be used to prove it.

Game G_0. The attacker can make q queries to an Oracle($\mathcal{O}$), and $\mathcal{O}$ responds the HSS-MAC$_i$. The attacker wins the game if the MAC is accepted when one of the queries is accepted.

$$\Pr[\mathcal{A}\ wins\ in\ G_0] \leq \epsilon \tag{4.13}$$

Game G_1. Compared with G_0, a given determined function is chosen instead of a random PRF. Moreover, each PRF is (q, T, ϵ) - pseudorandom. It holds that

$$\Pr[\mathcal{A}\ wins\ in\ G_0] \leq \Pr[\mathcal{A}\ wins\ in\ G_1] + \epsilon \tag{4.14}$$

Game G_2. Compared with G_1, the difference is that the $\mathcal{O}$ can record the previous input queries and output results. Once a new query is equal to the recorded pairs, the output will be the same. Based on this,

$$\Pr[\mathcal{A}\ wins\ in\ G_1] \leq \Pr[\mathcal{A}\ wins\ in\ G_2] \tag{4.15}$$

Game G_3. Compared with G_2, the difference is that the $\mathcal{O}$ does not contain the record function. There are q queries $Y_1, Y_2, \ldots, Y_q$, and each of them is distinct. In this situation, G_2 and G_3 proceed identically unless F happens, which is $Y_i = Y_j$ for some $i \neq j$. $G_2 \wedge \neg F \iff G_3 \wedge \neg F$. According to the difference lemma, we can deduce

$$|\Pr[G_2] - \Pr[G_3]| \leq \Pr[F] \tag{4.16}$$

F is an union of $\binom{q}{2}$ events, we can prove that $\Pr[Y_i = Y_j] = 2^{-L}$, thus we can derive that,

$$\Pr[F] \leq \frac{Q^2}{2^L} \tag{4.17}$$

L is the length of HSS-MAC. Therefore, we can make a conclusion that

$$\Pr[\mathcal{A}\ winsG_3] \leq \epsilon + \frac{Q^2}{2^L} \tag{4.18}$$

which is negligible. □

4.4 Energy Optimization in PV-MAC

In this section, we discuss the optimization problem in the PV-MAC scheme in offline verification and online verification.

In both situations, multiple independent data sources transmit the messages. Consequently, there are differences in both the quantity and size of cells. In offline verification, all the data sources transmit the messages at the same timeslot. The FN only needs to store one optimized solution if its workload does not change. However, data sources have different transmission frequencies in online verification. For example, the data about temperature will be shared frequently, while the data about some emergencies (e.g., fire alarms) will not be shared as frequently as that of temperature. Therefore, the number of cells verified by the FN will be altered as the amount of incoming messages changes.

In fog computing, the FN has great computation ability. Thus, we can partially offload the high computation cost of verifying the message integrity to the FN. As a consequence, the energy consumption of DU can be minimized. There are some limitations as follows,

1. The number of verified cells in one message should be smaller than or equal to the number of cells in the same message.

2. The number of verified cells should be large enough so that the FN can recover the master secret in the HSS-MAC scheme.

3. The workload of the FN cannot exceed its computation ability, which is determined by the hardware.

Table 4.2 shows the notations in the optimization problem, which is formulated for-

Table 4.2: Variables List for Optimization

Parameter	**Instruction**
i, m	Index, the number of message
$W_{FN}^{\max}$	The upper-bound workload of the FN
L	The number of levels in the message
B_i	The cell size of message i
t_j	The threshold of HSS in level j
N_i	The number of cells in message i
w^j	The weight of levels j
n_j	The number of cells in level j
τ_i	The verified cells in the FN in message i

mally as,

$$\min_{|\tau_i|} \quad E(\sum_{i=1}^{m}(n_i - |\tau_i|)) \tag{4.19a}$$

$$\textbf{s.t.} \quad \forall i, \quad |\tau_i| \le N_i \tag{4.19b}$$

$$\sum_{i=1}^{m} B|\tau_i| \le W_{FN}^{\max} \tag{4.19c}$$

$$\forall i, HSS_r(\tau_i) = s \tag{4.19d}$$

$$|\tau_i| \text{ is integer} \tag{4.19e}$$

4.4.1 The Level Weight Generation

Eq. 4.19d means that the FN should select enough cells to recover the master secret. Then, the FN sets the weight (w^i) for each level to reach the optimization targets. We use T to represent the threshold weight to recover the master secret.

$$\begin{aligned} w^1 t_1 &\geq T \\ \forall l \leq L, \quad \sum_{j=1}^{l-1} w^j (t_j - 1) + w^l t_l &\geq T \\ \forall l \leq L, \quad \sum_{j=1}^{l-1} w^j (t_j - 1) + w^l (t_l - 2) + \sum_{i=l+1}^{L} w^j n_j &< T \end{aligned}$$

According to the conditions above, we can derive the relationships,

$$\forall 1 \leq l \leq L, \quad w^l > \sum_{j=l+1}^{L} w^j n_j / 2 \tag{4.20}$$

Suppose the weight of the lowest level (w^L) is 1. The FN can derive the rest level weight and the threshold weight. Consequently, Eq. 4.19d can be represented as,

$$\forall i, \forall l \leq L_i, \quad \sum_{j=1}^{l} w_i^j |\tau_i^j| \geq T_i \tag{4.21}$$

Insides, L_i is the number of levels in message i. w_i^j is the weight of level j in message i, and $|\tau_i^j|$ is the number of verified cells in level j in message i. Apparently, $\sum_{j=1}^{l} |\tau_i^j| = |\tau_i|$. T_i is the threshold weight of message i.

In summary, since B_i, N_i, w_i^j, and T_i are stable, solving the optimization is simple. The Eq. 4.19b, Eq. 4.19c and Eq. 4.19d are linear conditions. In the meantime, Eq. 4.19e shows that the optimization is an integer-based optimization problem. Consequently, we can apply Mixed-Integer Linear Programming [57] to optimize energy consumption for the DU. Chapter 4.6.6 shows the optimization results in two situations.

4.5 Improvement on PV-MAC

In this subsection, we propose an improvement on PV-MAC. Recall that in both SS-MAC and HSS-MAC, the subshares should be transmitted with the messages. Therefore, the communication cost enlarges.

In Shamir secret sharing, the dealer not only generates subshares for every participator but also generates the polynomial coefficients (e.g., $a_1 \dots a_{t-1}$). In the meantime, the Lagrange interpolation can reconstruct the polynomial. Therefore, instead of randomly generating the coefficients in the SS-MAC scheme, we can use the $t-1$ subshares of the next message to represent the coefficients.

$$f_{m+1}(x) = s + f_m(x_{m,1})x + \ldots + f_m(x_{m,(t-1)})x^{t-1} \bmod p \tag{4.22}$$

$f_m(x_{m,1}), \cdots, f_m(x_{m,t-1})$ are the subshares of the message m. In this situation, the data source only delivers $n-(t-1)$ subshares to the FN. Moreover, if $n-(t-1) < t$, the attacker will not get enough subshares to reconstruct the secret according to the security in secret sharing. After that, the substitution attacks fail.

As much, we can apply this structure to PV-MAC. Formally, it is called the Partial Verification Chained Message Authentication Code (PV-CMAC). According to Chapter 2.2.2, only the highest level polynomial and its coefficients must be recovered. Suppose there are two levels with $(t_1, n_1), (t_2, n_2)$ and the shadows are $(x_1, f_1(x_1)), ..., (x_{n_1}, f_1(x_{n_1})), ..., (x_{n_2}, f_2(x_{n_2}))$, only the $t_1 - 1$ shadows can be used as the coefficient of the polynomial for the next message. Consequently, the growth of time consumption and the decrease of communication cost is relevant to t_1. We give a concrete example to show how PV-CMAC works.

Example 6. *Suppose there are two messages: one is* $(1, 2, 3, 4, 5, 6, 7, 8)$*, and the other is* $(8, 7, 6, 5, 4, 3, 2, 1)$*. Each of them is divided into two levels, like Figure 4.4. The threshold in the two levels is 3. The prime for secret sharing is 97, and the secret for the first message is 50.*

For the first message, the HSS follows the same process as Figure 4.4. However, in the HSS scheme for the second message, the coefficients of $f_1(x)$ are replaced by the subshares

$$f_1(x) = 50 + 33x + 3x^2 \bmod 97$$
$$f_2(x) = 84 + 15x + 28x^2 \bmod 97$$

For the first message, the processed message is

$$(1, 2, 3, 4, 5, 6, 7, 8, 57, 1, 85, 40, 28, 49)$$

The first two subshares are hidden in the polynomials. Thanks to the perfect security in secret sharing, the attacker cannot recover the secret or reconstruct the HSS-CMAC.

The subshares of the second message is

$$(19, 34, 66, 18, 10, 90, 32, 30)$$

The data sources can send the subshares of the final message through a secret channel before the message transmission. The fog node verifies the messages in reverse order from the incoming messages.

Compared with the original situations, PV-CMAC can save the communication cost. However, storing the coefficients consumes the workload in the FN, and recovering all the coefficients costs more time rather than only recovering the secret.

4.6 Experiment Result

In this section, we use some experimental results to demonstrate the performance of our proposed schemes. We compare the performance of PV-MAC, PV-SS-MAC, PV-CMAC, and concatenating key-hashed MAC (CHMAC) in different situations. The CHMAC means the DS generates a MAC for each cell using the HMAC scheme and concatenates them into one final MAC. PV-SS-MAC combines the SS-MAC [73] and the SPMAC scheme. The third experiment focuses on workload allocation in the FN for offline and online verification.

The prime, key, generator, and nonce for SPMAC are 160 bits. The key for HSS-MAC and SS-MAC is 1024 bits. In the CHMAC schemes, we apply SHA256 for the HMAC. For the energy model, we consider $E = \tau^2 + 2.4\tau + 3.5$ in [62].

To evaluate the performance, we apply two datasets. The first one is the synthetic dataset, which contains randomly generated messages with the same cell size and length. The second one is the real-life dataset,'Room Occupancy detection data', which includes temperature, Humidity, Light, CO2, and Humidity Ratio [15].

The code of both FN and DU is implemented in Python3. Since the IoT devices have a limited workload, we set up the IoT devices on a Raspberry Pi with 1 GB RAM and 1 CPU core. The FN is set up on a Virtual Machine (VM) with 4 GB RAM and 4 CPU cores.

4.6.1 Time Consumption of PV-MAC under Different Cell Size

In this part, we discuss how the cell size affects the performance of PV-MAC. One message contains 400 cells with varying cell sizes ranging from 1 bit to 32 bit. A ((21, 10, 4, 34, 29, 32, 15, 32, 8, 25), 400) HSS scheme is applied in PV-MAC, and the number of cells in each level is 40. The threshold of each level is randomly generated. In Figure 4.5 and 4.6, the time for PV-MAC contains that for HSS-MAC.

In Figure 4.5, the time consumption of PV-MAC grows when the cell size increases. The time consumption of HSS-MAC in PV-MAC decreases. The cell size has a

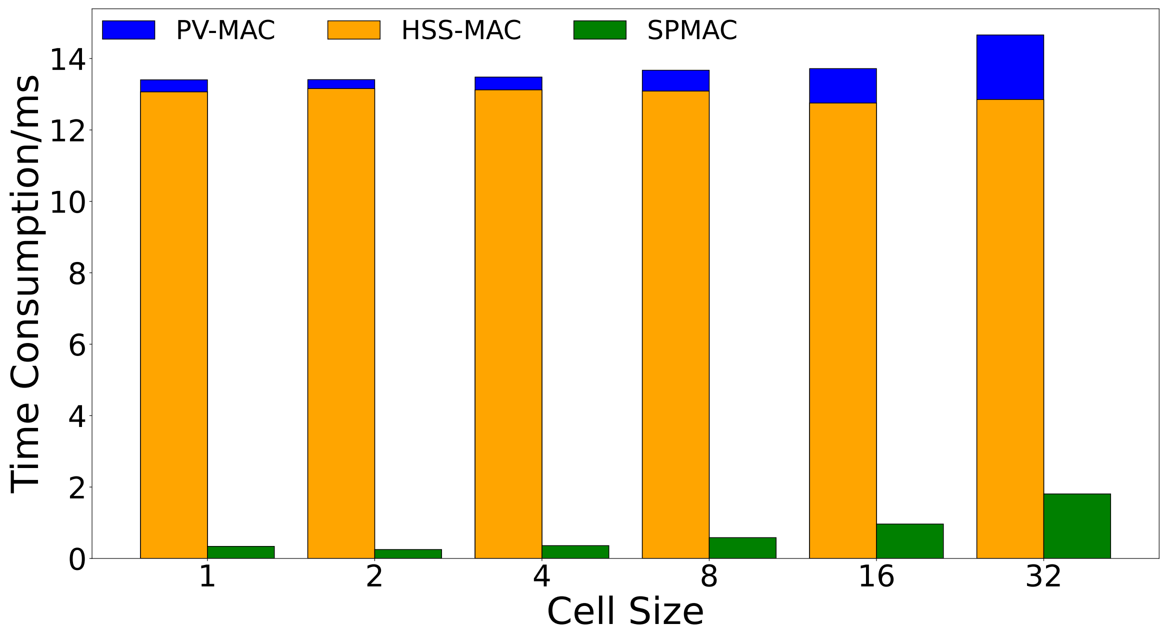


Figure 4.5: Time Consumption under The Different Cell Size

negligible effect on HSS-MAC but more on SPMAC verification.

4.6.2 Time Consumption of PV-MAC under The Number of Levels

In this part, we look at how the verified cells coming from different levels affect the performance of PV-MAC. There are 400 cells in the message, and we divided them into ten levels. The FN randomly chooses 200 cells to verify. A $((80, 10, 16, 6, 19, 15, 4, 2, 10, 14), 400)$ HSS scheme is applied in PV-MAC.

Based on Figure 4.6, the time consumption gradually decreases when the verified cells come from more levels. In Chapter 4.3.1, we give the equation of the time consumption of HSS-MAC. When the verified cells come from different levels (i.e., $l_1 < l_2$) and the number of verified cells does not change, $O(\sum_{i=1}^{l_1} t_i \log^2(t_i))$ must be larger than $O(\sum_{i=1}^{l_2} t_i \log^2(t_i))$.

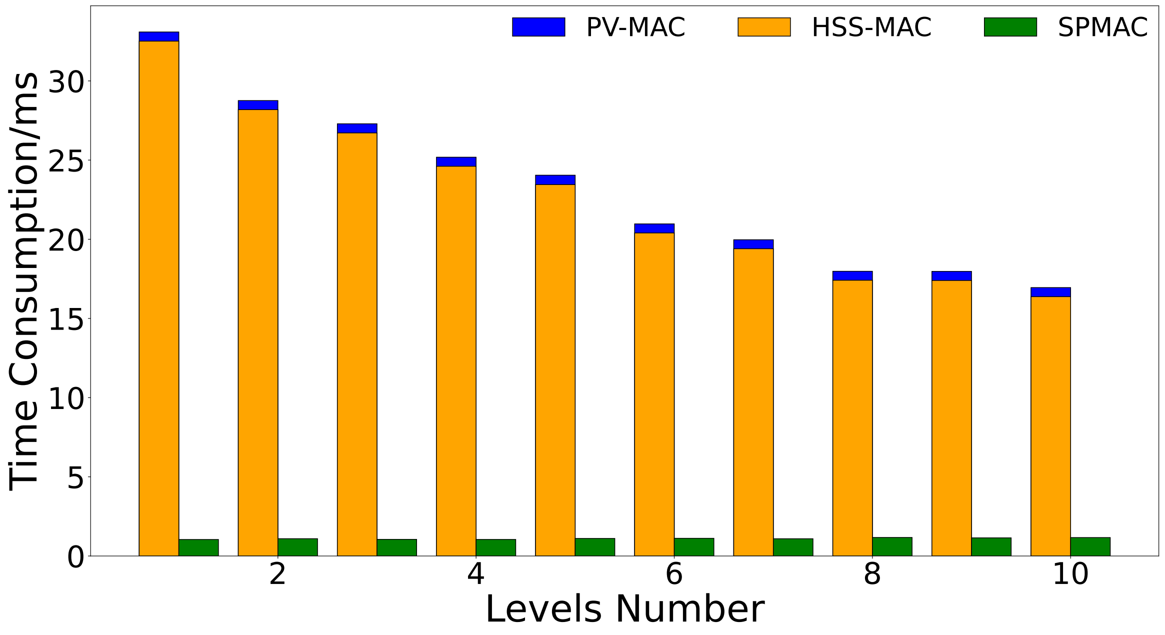


Figure 4.6: Time Consumption under The Verified Cells from Different Levels

4.6.3 Comparison between PV-SS-MAC and PV-MAC

In this part, we compare the time consumption of PV-SS-MAC and PV-MAC when the number of verified cells rises from 178 to 328. A $((23, 26, 18, 18, 31, 16, 18, 5, 3, 20), 400)$ HSS scheme and $(178, 400)$-secret sharing.

According to Figure 4.7, we can observe that the time consumption of the DU decreases. The reason is that the number of cells verified by DU declines in both schemes. Furthermore, the time consumption of PV-SS-MAC is more significant than that of PV-MAC, which proves our deduction in Chapter 4.3.1. In addition, the time consumption of PV-SS-MAC increases faster than that of PV-MAC. Since the second derivation of Eq. 4.11 is greater than 0, the increasing rate of time consumption of PV-SS-MAC becomes larger.

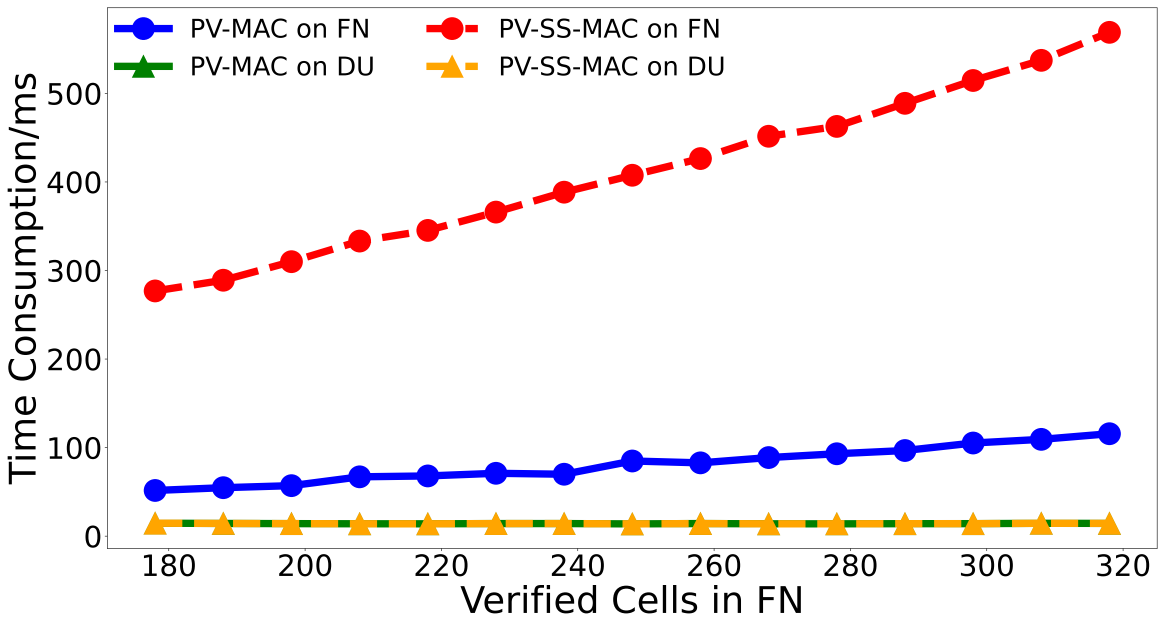


Figure 4.7: Time consumption of PV-SS-MAC and PV-MAC

Table 4.3: Time Consumption and Communication Cost of The Three Schemes

	PV-SS-MAC	PV-MAC	CHMAC
Time for FN/ms	8592.232	2132.176	563.071
Time for DU/ms	355.102	340.896	394.134
Communication Cost/kbit	1964.597	1964.646	25500

4.6.4 Performance of PV-SS-MAC, PV-MAC, and CHMAC

In this subsection, we compare the time consumption and communication cost among PV-SS-MAC, PV-MAC, and CHMAC. The FN verifies 234 cells.

Table 4.3 shows the three schemes' accumulated time consumption and communication cost. Although concatenating HMAC consumes the least time among the three schemes, the communication cost is 12.98 times larger than the other two. Therefore, it is not suitable for our proposed scenario. Compared with PV-SS-MAC, PV-MAC

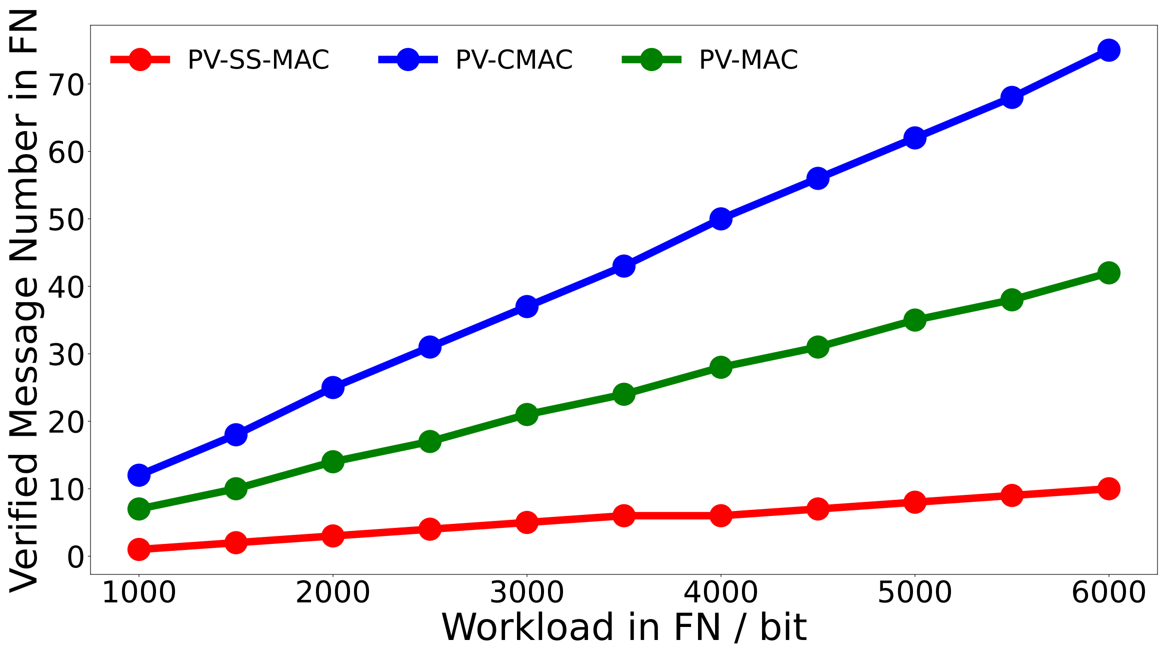


Figure 4.8: Workload Allocation for Three Schemes

consumes less time, which has been explained in Chapter 4.6.3.

4.6.5 Workload Allocation PV-SS-MAC, PV-MAC, and PV-CMAC

In this subsection, we compare the maximized number of messages verified by the FN at the same time when the workload changes. There are 50 cells in one message, and it has been divided into six levels. The threshold of SS-MAC is 28, which is equal to the sum of thresholds in each level of HSS.

According to Figure 4.8, the number of messages verified by FN in PV-SS-MAC is the least among the three schemes. The reason is that the FN should verify at least 28 cells. However, at least the cells in the first levels should be verified in PV-MAC, which is smaller than that in PV-SS-MAC. In PV-CMAC, the FN should save the coefficients from the last message. Compared with the PV-MAC, the workload for verification decreases.

Table 4.4: The Number of Cells and The Cell Size of Incoming Message

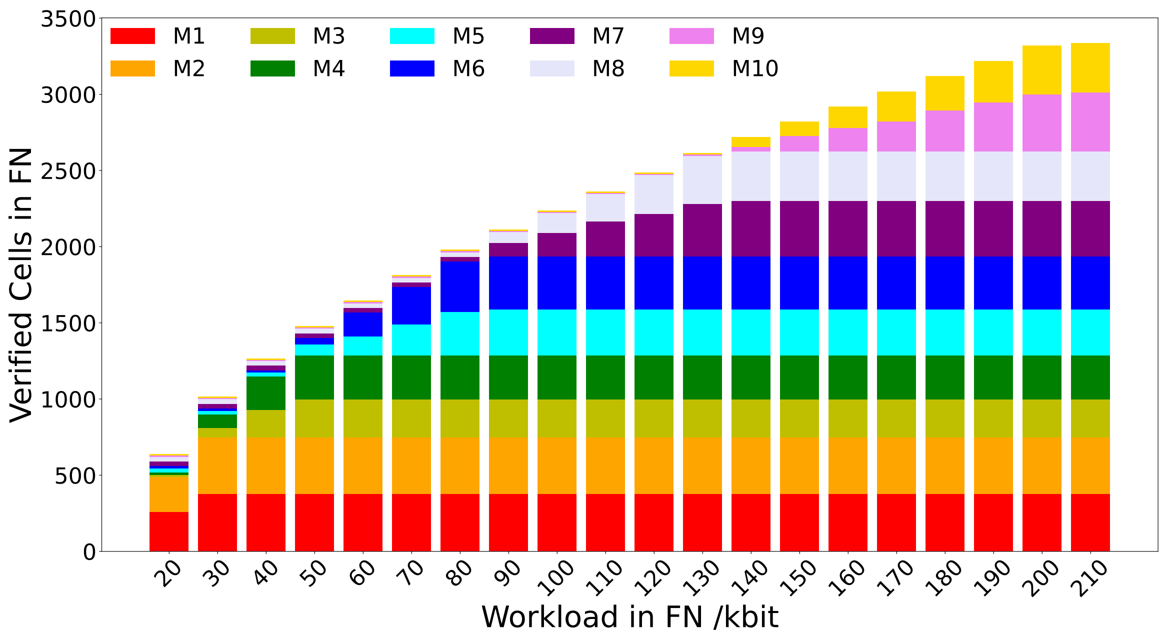
	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
Number of Cells	378	371	249	288	304	346	365	324	388	322
Size of Cell/bit	20	20	40	40	60	60	80	80	100	100

4.6.6 Performance of PV-MAC in Offline and Online Verification

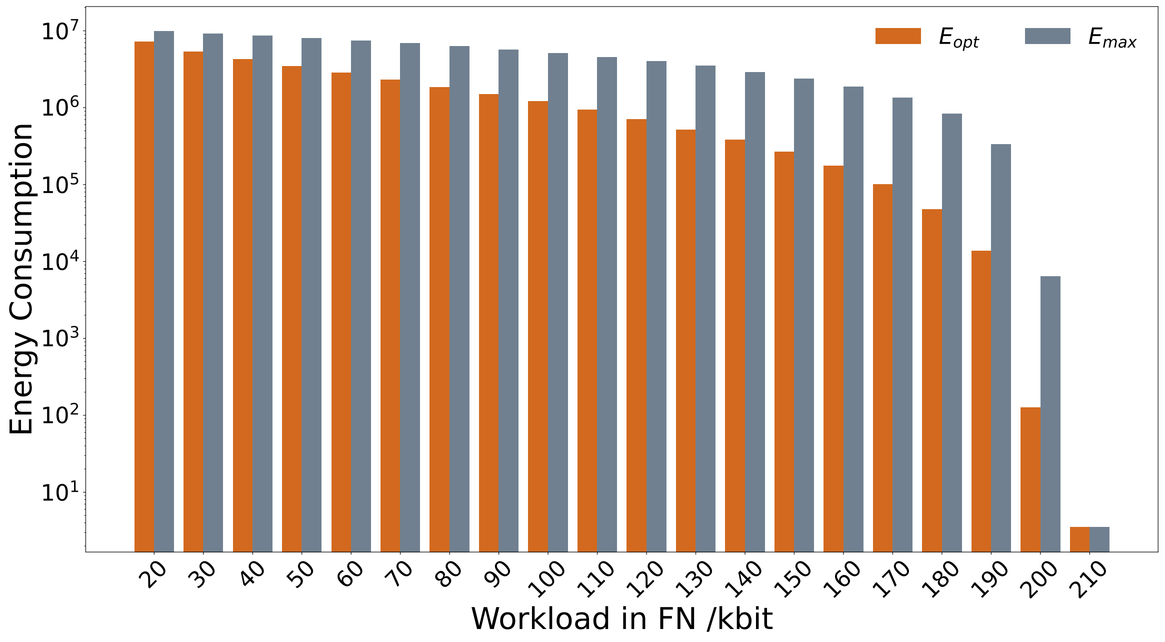
In this subsection, we demonstrate the operation of PV-MAC in two real-world scenarios: offline and online verification. We randomly divide the practical datasets into several pieces, and the number of cells and the size of cells are given in Table 4.4. We compare the energy consumption of DU with two different verification models. One is the optimized results (E_{opt}). The other model is that the FN first chooses the message with large cell sizes for verification (E_{max}) We show how the number of verified cells in FN varies as the upper-bound workload of FN increases. The workload increases from 20kbit to 210kbit.

According to the energy model, the growth of the number of cells verified by DU leads to an increase in energy. Our objective is to minimize the energy consumption in DU. Thus, the number of cells verified by FN should be maximized. Therefore, the message with a small cell size is entirely verified by FN if the workload is large enough. According to Figure 4.9(a), when the workload of FN is 20kbit, the number of cells in M_1 and M_2 is larger than the rest of the messages. With the increases in the workload, the FN can verify more cells. Accordingly, the message with a small cell size can occupy more workload of FN. When the workload of FN reaches 210kbit, it can verify all the messages.

In Figure 4.9(b), we can find out that when the workload of FN is limited, it cannot



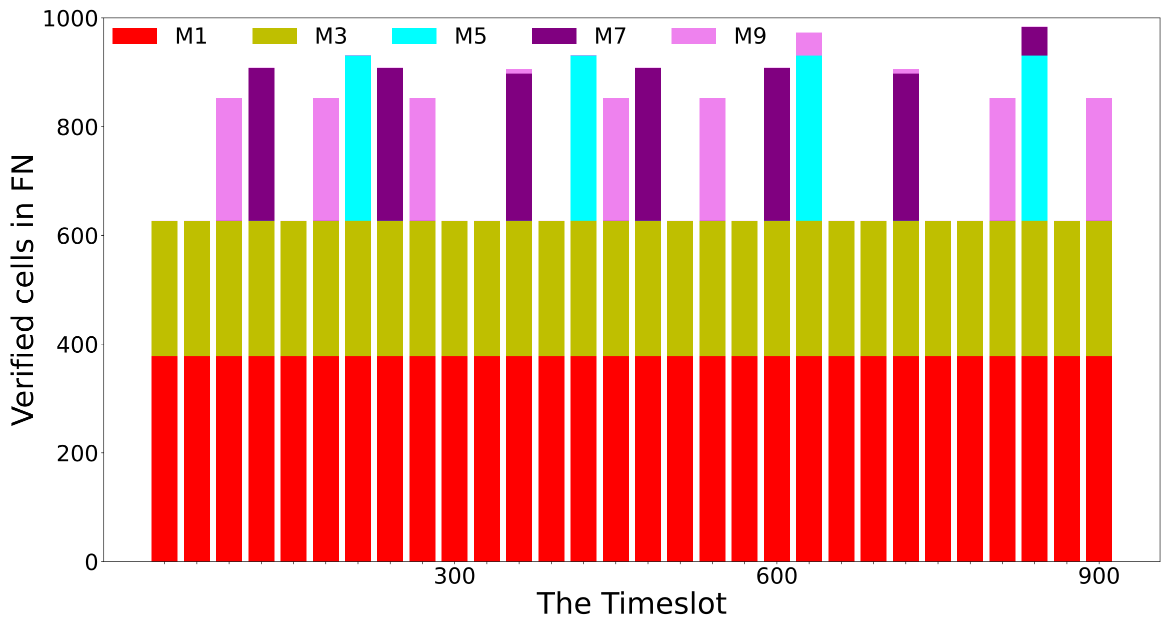
(a) The Number of Verified Cells under The Upper-bound Workload of FN



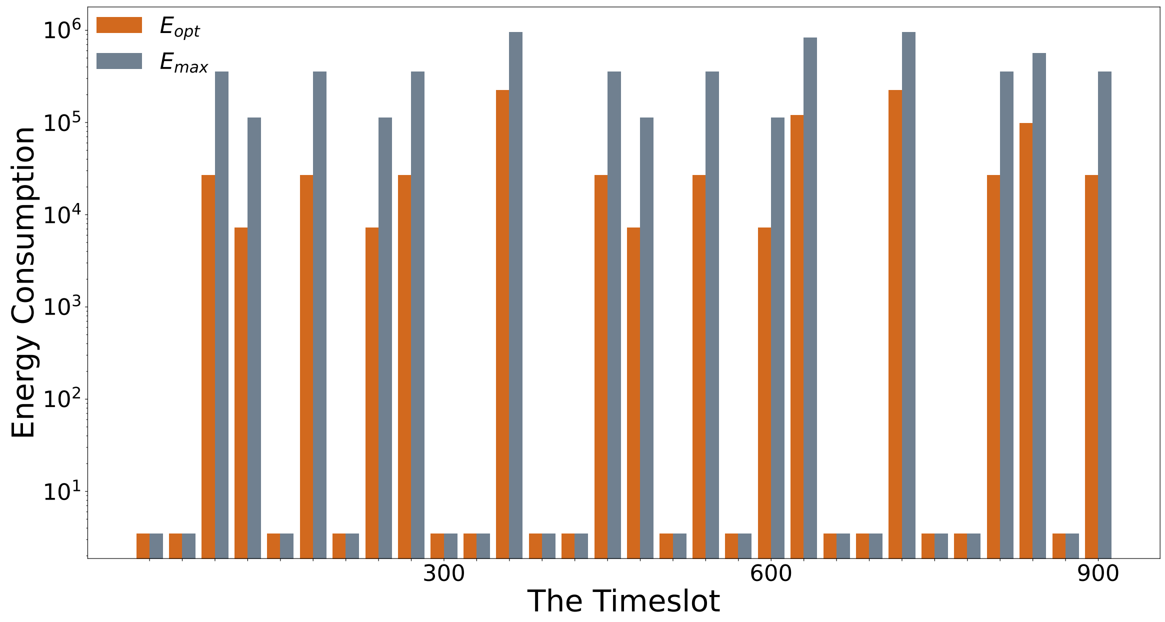
(b) The Semi-Plot of The Energy Consumption of DU under The Upper-bound Workload of FN

Figure 4.9: Performance of Offline Verification

verify too many cells. Thus, the energy consumption is pretty significant. The larger the workload of FN is, the less energy the DU consumes. Moreover, the difference



(a) The Number of Verified Cells under The Timeslot



(b) The Semi-Plot of The Energy Consumption of DU under The Timeslot

Figure 4.10: Performance of Offline Verification

between the two energy models is large when the workload increases from 20kbits to 200kbits. Since in the E_{max} model, the FN first chooses the messages with large cell sizes to verify. Thus, the number of verified cells in FN is small, and the DU verifies more cells than that in the E_{opt} model. However, when the workload of FN is 210kbits, the DU does not need to verify the message. Consequently, the energy consumption in both models is the same.

In the context of online verification, we present how the number of cells verified by FN changes. Formally, as the frequency of incoming messages varies, the workload of the FN necessitates periodic adaptation. In this experiment, there are five DS, and the messages follow like M_1, M_3, M_5, M_7, M_9 in Table 4.4. Each DS transmits one message every 2, 3, 7, 8, and 9 seconds, respectively. The workload in the FN is 40kbit. We illustrate the performance of workload allocation every 30 seconds.

Figure 4.10(a) shows that the number of cells verified by the FN is affected by time. Since the number of incoming messages is different in different timeslots, the number of cells verified by FN changes periodically. Furthermore, at the same timeslot, the FN first verifies the message with a small cell size, as explained in Chapter 4.6.6.

Moreover, if the FN cannot verify all the cells in the incoming messages, the energy consumption of DU in E_{opt} is smaller than that in E_{max} in Figure 4.10(b). Energy consumption also modifies systematically with the number of incoming messages changing.

4.7 Summary

In this chapter, we focus on the subset verification problem in a fog-based IIoT scenario. With IoT devices widely deployed in the Industrial scenario, they transmit a large number of messages to the data user. The data user cannot verify all the incoming messages concurrently. Consequently, leveraging a dependable fog node can assist in alleviating the computational burden associated with verifying message integrity. To achieve our objective, we propose the Partial Verification Message Authentication Code (PV-MAC). The data source can generate the MAC for the whole message.

Then, the fog node can randomly choose the cells in the messages and verify their integrity. After that, the data user can verify the rest of the cells. Moreover, the fog node can optimally choose the maximum number of cells it can verify according to its computation ability in order to decrease the energy consumption of the data user. Theoretical and empirical analyses validate the efficiency of our scheme, with marginal increases observed in communication costs.

Our proposed scheme holds applicability across various domains. For instance, in a machine learning scenario, the datasets are large, and their integrity is essential. Additionally, the datasets are from different data sources. Thus, their size and frequency of transmission are different. In this situation, a fog node can help verify a subset of the datasets and train them according to its workload. However, it trains the model so that it cannot exhaust the whole workload for verification. Therefore, the data user verifies the rest of the subsets and sends them back to the fog node for additional training.

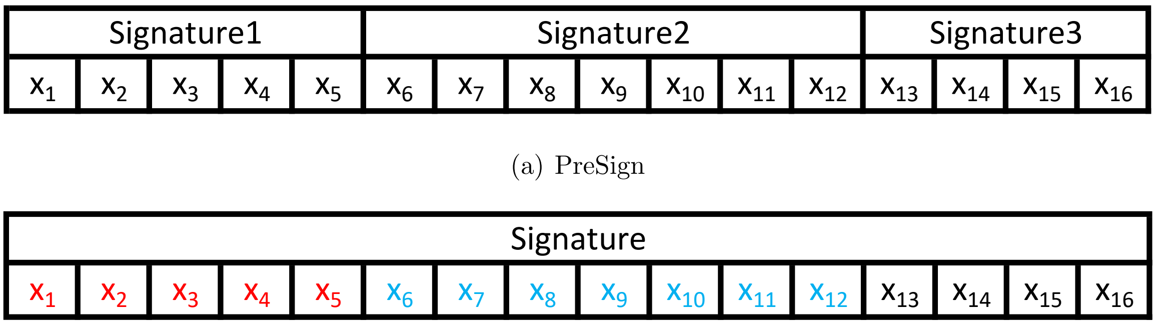
Chapter 5

Subset Integrity among Multiple Entities

In this chapter, we solve a more general problem in the IoT scenario, which is subset integrity among multiple entities. Compared with the problem in Chapter 4, the problem is more complicated, which happens in the collaborative edge computing scenario, where several edge nodes cooperate to finish the verification tasks. This process is similar to the multi-hop computation offloading. However, subset verification is more complex than multi-hop computation offloading [44]. The participants can only verify a subset of the message, while proof of integrity (e.g., signature) is generated according to the whole message. Consequently, there should be a relation between the signatures of a subset of the message and the entire message. Besides the subset integrity problem, another problem is inside: Subset Aggregation. The messages can be aggregated even if different private keys sign them. Generally, the number of signatures becomes large, leading to a high communication cost. [34] proposed aggregated signature by combining several signatures. It requires the utilization of the same private key to generate all the signatures. One of the applications is the smart city. Each IoT device with limited computation ability is deployed in the city, and different owners own the IoT devices within the smart city scenario. As a result, they have distinct private keys to sign the message.

Suppose one message contains several cells (e.g., $\mathbf{x} = (x_1, x_2 \ldots)$), there are three kinds of solutions for subset verification. First, the signer signs a subset of the message (e.g., PreSign). According to Figure 5.1(a), the signer gave three signatures previously, and the verifier can check the integrity of predetermined subsets (e.g., x_1 to x_5). However, the verifier cannot autonomously choose the cells for verification (e.g., x_3 to x_5). Second, the signer generates the signature for each cell and concatenate them together. Therefore, the verifier can randomly pick several cells for verification. However, the communication overhead is severe when the number of cells is enormous since the size of the signature is proportional to the number of cells. The final solution is based on threshold signature [79]. Before signing the message, the signer generates a secret value. Then, according to the secret sharing scheme [33], the signer generates the subshares for each cell and transmits them with the signature to the verifier. Upon receiving them, the verifier selects a group of cells and their subshares to recover the secret, whose number should be larger or equal to the threshold (e.g., x_1 to x_6 or x_{12} to x_{16}). However, the time consumption is significant in the threshold signature. For example, if the verifier chooses t cells to check the integrity, the time consumption is $O(t \log^2(t))$ [6]. In the meantime, the verifier cannot guarantee their integrity if the unverified number of cells is smaller than the threshold. In summary, Table 5.1 shows the drawbacks of the three possible solutions. Consequently, they are inappropriate for collaborative edge computing scenarios in smart cities.

To address the problem, we propose Elliptic Curve based Multi-hop Sanitizable Signature (ECMSS). The signer generates only one signature, and the verifiers can randomly select a subset of the message for verification by themselves. ECMSS is based on sanitizable signature [7], which allows the authenticated party (i.e., delegator) to change parts of the cells. Consequently, the verifier can ensure the integrity of the rest cells in the message. The property of the sanitizable signature is that it can


Signature1
Signature2
Signature3
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16
Signature
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16

(a) PreSign

(b) Threshold Signature

Figure 5.1: Example of Subset Verification

protect sensitive information from exposure. In other words, the edge node can verify the subset of the message, which is not sensitive. The property can also be applied to subset verification. The verifier can check the integrity of the subset he wants to verify without knowing the subset he does not need to verify. The ECMSS has some practical applications.

- **Big data analysis** [61]: Some participants have limited computation resources (e.g., roadside units and robot swarms) and must finish plenty of big data analysis work. They can assign the calculation jobs to the adjacent devices. Before the analysis, they can ensure a subset of data integrity by verifying the signature based on ECMSS.
- **Distributed edge learning** [109]: Several edge nodes can cooperate to train large datasets. Before the training, the integrity of the datasets is essential. With the ECMSS scheme, the edge nodes can verify and train a subset of data according to their computation ability.

To summarize, our main contributions are as follows.

- We propose a lightweight scheme called ECMSS for subset aggregation. The edge node can concatenate the incoming messages and aggregate the signatures even if different private signing keys generate the signatures.

Table 5.1: Comparison of Solutions for Message Subset Integrity

Name	PreSign	Concatenating Signature	Threshold Signature
Subset Aggregation	×	✓	×
Subset Verification	✓	✓	✓
Autonomous Verification	×	✓	✓
Communication Efficiency	✓	×	×
Time Efficiency	✓	×	×

- The ECMSS scheme can achieve subset verification, which means the edge node can randomly select a subset of the message for verification.
- We theoretically analyze our proposed scheme to show its security. The experiments show that our proposed scheme can reach high efficiency and low communication overhead.

The organization of the chapter is as follows. Chapter 5.1 introduces the network model and defines the problem we discuss. Chapter 5.2 demonstrates the ECMSS. Chapter 5.3 provides the optimization solution, and Chapter 5.4 analyzes the security. In Chapter 5.5, we use some experiments to show the performance of ECMSS. Chapter 5.6 summarizes this chapter.

5.1 Problem Definition and System Model

5.1.1 System and Threat Model

Edge computing aims to leverage the computation ability of the edge node to save transmission delay. Unfortunately, the edge node confronts a bottleneck in computation ability. The cost of developing and purchasing edge nodes with more extraordinary computation ability is large [9]. Therefore, the economical and efficient solution is for several edge nodes to cooperate to finish the tasks.

Our collaborative edge computing model consists of four distinct entities. First, the data sources (DS) are a group of IoT devices deployed in the smart city. They gather data on temperature, humidity, and traffic. Subsequently, utilizing their individual private keys, they generate unique signatures. Adjacent to the DS, the Gathering Edge Node (gEN) helps concatenate the incoming messages and aggregate the corresponding signatures. This node serves the purpose of streamlining communication and data processing within the system. The data user (DU) is another IoT device requiring messages. However, he cannot verify all the incoming messages due to his computation ability. A collective of Verifying Edge Nodes (vEN) collaboratively assists in verifying the incoming messages and sending corresponding results to the DU. Figure 5.2 shows the communication model with one gEN and two vENs.

The communication model consists of two sides. The DS and gEN are the sender side. The structure is Cluster-based Edge Computing [80]. Generally, clustering is a network topology management that improves the efficiency of the network. The gEN is the cluster head, and the DS is the cluster member according to their location. The gEN gathers the messages and signatures from the DS. The vEN and the DU are the verifier side. The structure is based on offloading and feedback structure [106][42]. Once the edge node accepts the computation tasks, he can offload parts of them to the next edge node (i.e., the offspring node). For instance, vEN$_1$ executes parts of the mission and transmits the rest parts to vEN$_2$. In the feedback phase, all the edge nodes participating in the computation give the computing results to the user. Specifically, every vEN$_{i+1}$ transmits the computing result to vEN$_i$ (i.e., the ancestor node). Furthermore, the relationship between the two vENs is dynamic. For example, vEN$_1$ is the ancestor node of vEN$_2$ in the first task, while the relationship may be reversed in the second task. The advantage is that each vEN can choose the number of tasks by himself rather than be allocated by others. Moreover, we consider that a reliable device-to-device (D2D) communication [40] is applied among the vENs. Furthermore, as with existing integrity assurance schemes, our scheme does not need to assume a homogeneous network [99]. In the same way, packet loss

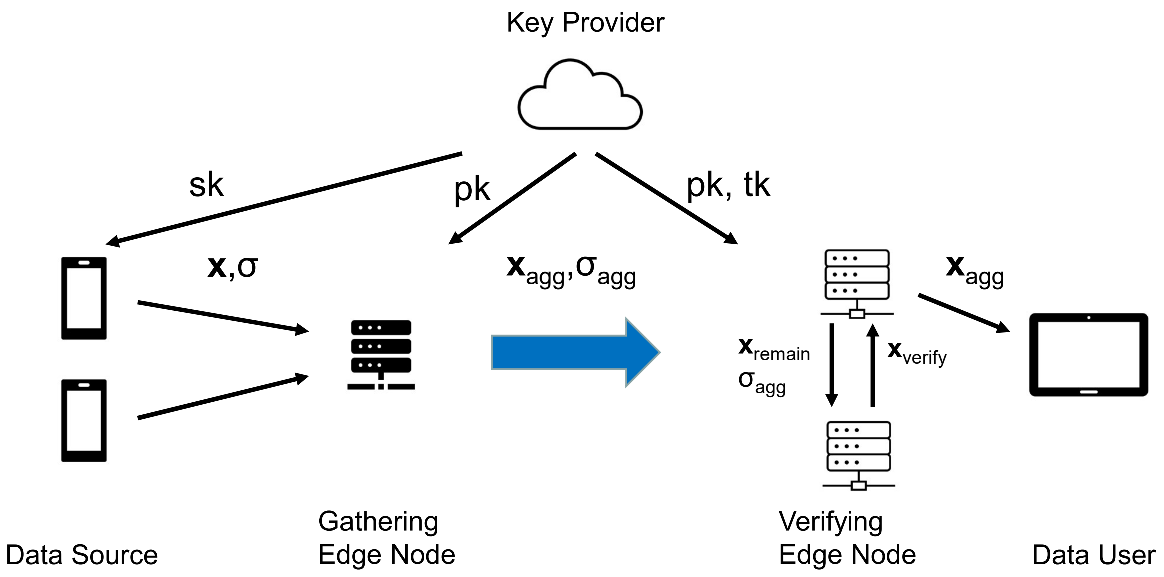


Figure 5.2: System Model

[2] or even timeouts may affect the performance of our scheme, as affecting other network services. Since then, we have not addressed this issue in this chapter.

The adversary model is shown as follows. The attacker can observe the signatures for a polynomial of numbers. He has infinite computation resources to forge the message. Specifically, the attacker will try to modify the message from $\mathbf{x}$ to $\mathbf{x}'$ without changing the signature. Consequently, the DU cannot use the message.

5.1.2 Problem Definition

Based on the communication model, we discuss two problems. First, the signature is generated by different IoT devices. As a consequence, how to aggregate all the signatures and messages with different keys is essential. Formally, we called it **subset aggregation**. Second, since one edge node cannot finish the verification, how to achieve subset integrity verification in collaborative edge computing is essential. Specifically, each edge node can **randomly** and **optimally** choose the random number of cells to verify before the DU according to its computation ability and energy.

Furthermore, the edge node can send cells that it has not verified to the next edge node. This problem is split into two parts: **subset verification** and **optimization problem**.

We consider that one message contains several cells (e.g., $\mathbf{x} = (x_1, x_2, \ldots, x_n)$). The definition of subset aggregation is as follows,

Definition 5. ***(Subset Aggregation)****: The gEN can aggregate the incoming signatures even if they are generated by different private signing keys.*

$$\begin{aligned}&\sigma_1 \leftarrow F(sk_1, \boldsymbol{x}_1) \textit{ and } \sigma_2 \leftarrow F(sk_2, \boldsymbol{x}_2),\\&\sigma_{agg} \leftarrow \sigma_1 \otimes \sigma_2\end{aligned}$$

$\mathbf{x}_1$ and $\mathbf{x}_2$ are the original message from DU_1 and DU_2, sk_1 and sk_2 are the different private keys for signature generation. σ_1 and σ_2 are the signatures of $\mathbf{x}_1$ and $\mathbf{x}_2$. $\otimes$ is the binary operation to aggregate the two signatures. F is the signature generation function. If a verifier wants to verify the message, he should use the public keys from DU_1 and DU_2.

The definition of Subset Verification is as follows:

Definition 6. ***(Subset Verification)****: The vEN can randomly choose a subset in one message for verification by himself.*

$$\forall \quad \boldsymbol{x}_v \subseteq \boldsymbol{x}, \quad 1 = VF(pk, \boldsymbol{x}_v, aux_v, \sigma)$$

$\mathbf{x}_v$ is the subset of $\mathbf{x}$, which is verified by vEN. pk is the public key for verification. VF is the verifying function. Since the signature is generated based on the whole message, the verifier cannot use a subset of the message to reconstruct the signature directly. Therefore, some auxiliary information helps the verifier to reconstruct the signature. Specifically, the aux_v cannot contain anything about the subset that the vEN does not need to verify.

The subset verification is similar to the computation offloading [82]. In this context, the optimization problem is essential. When many messages are coming, the verification tasks will be burdensome, demanding efficient optimization strategies to handle the increased workload. In this scheme, we want to minimize the energy consumption in the collaborative edge computing system according to Eq. 5.1.

$$\begin{aligned} \min \quad & E_{system} = E^{\text{verify}} + E^{\text{tran}} \\ \textbf{s.t.} \quad & \forall i, E_i(\tau_i) \leq E_i^{max} \\ & \forall i, W_i(\tau_i) \leq W_i^{max} \\ & \tau_i \quad is \quad integer \end{aligned} \tag{5.1}$$

Insides, E_{system} is the objective function, which is the energy consumption for both verification (E^{verify}) and transmission (E^{tran}). τ_i is the number of cells. E_i and W_i are the energy and workload of the i-th edge node. E_i^{max} and W_i^{max} are the upper-bound energy and workload of the i-th edge node. The E^{verify} is relevant to the number of cells for verification, while the number of transmissions is relevant to the network properties. The details will be given in Chapter 5.3.

5.2 Elliptic Curve based Multi-hop Sanitizable Signature

Generally, the PV-MAC schemes still need the client to verify a subset of messages, which is not suitable in the resource-constraint scenario. This section aims to present the Elliptic Curve based Multi-hop Sanitizable Signature (ECMSS) within the context of collaborative edge computing. Specifically, our objective is to address the challenge when an individual edge node does not have enough computation ability to verify the incoming message integrity. Without taking privacy into account, the sanitizable signature can assist in verifying a subset of the message without knowing the remaining parts. This approach significantly enhances efficiency in the verification process.

5.2.1 Elliptic Curve Based Chameleon Hash

In this part, we review an Elliptic Curve Based Chameleon Hash (ECCH), which is the primary component of ECMSS. Eq. 5.2 shows the construction of ECCH.

$$ECCH(x) = xG + rK_0 \tag{5.2}$$

Inside, G is a random generator in the elliptic cyclic group. $K_0 = kG$, and k is the trapdoor key. r is the random number. If the delegator wants to find out the collision, he should find out a pair of (x', r') so that,

$$x + kr = x' + kr' \tag{5.3}$$

The vEN with the trapdoor key can quickly discover the collision. Conversely, it is difficult for the third party who does not have the trapdoor key to discover the collision due to the Elliptic Curve Discrete Logarithm Problem (ECDLP). The ECCH has the same properties as the traditional chameleon hash. Moreover, similar to Eq. 5.3, the vENs can change the trapdoor key as Eq. 5.4 in the multi-hop verification scenarios.

$$x + kr = x' + k'r' \tag{5.4}$$

The new trapdoor key is k'. Once a vEN does not need to be involved in the verification, it cannot sanitize any subset of the message. In its view, the ECCH becomes a one-way hash function, which has been introduced in Chapter 2.2.3.

Table 5.2: Variables List for ECMSS

Parameter	**Instruction**
i, j, h	The index of the gEN, vEN, and IoT nodes
M, N	The number of gEN and vEN
T_i	The number of IoT belongs to the i-th gEN
$\mathbf{x}_h$	The message from the h-th IoT nodes.
$\bar{\mathbf{x}}_j$	The message verified by j-th vEN.
$\mathbf{x}$	The final message
σ_h	The signature from the h-th IoT node
σ_i	The aggregated signature in the i-th gEN
σ_{agg}	The aggregated signature
G	The generator of the cyclic group
s_h, S_h	The private and public signing keys for the h-th IoT nodes
r_i, r_0	The random nonce of the i-th gEN and all the vEN
R_j	The sanitization result of the j-th vEN
H	The one-way hashing
k_0, K_0	The private and public sanitization keys for all gEN
k_j, K_j	The private and public sanitization keys for j-th vEN
$rem(\bar{\mathbf{x}}_j)$	The cells which are not verified by j-th vEN
$att(\mathbf{x})$	The attributes of $\mathbf{x}$

5.2.2 Elliptic Curve Based Multi-hop Sanitizable Signature

In this part, we introduce the Elliptic Curve based Multi-hop sanitizable signature (ECMSS). 5.2 shows the notation of ECMSS. There are five steps in the ECMSS scheme.

- **Setup**(1^λ): The Setup algorithm is a probabilistic algorithm that takes as input a parameter λ. It generates two groups of keys. One is for signing (S_h, s_h), and the other is for sanitizing $((K_j, k_j))$.

$$(S_h, s_h), (K_j, k_j) \leftarrow 1^\lambda$$

- **Sign**: For the h-th IoT node, it takes the message $\mathbf{x}_h$, a private signing key s_h, a public sanitization key K_0 and the random integer r as input, and outputs the signature.

$$\sigma_h \leftarrow Sign(\mathbf{x}_h, r_h, s_h, K_0)$$

- **Aggregate**: For the i-th gEN, it takes the received signatures and the public keys as input and outputs the aggregated signature. Agg is the aggregation function

$$\sigma_i \leftarrow Agg(\forall \sigma_h)$$
$$S_i \leftarrow Agg(\forall \S_h)$$

- **Subset Sanitize**: For the j-th vEN, it takes a subset of message $\bar{\mathbf{x}}_j$, a subset substituted message $\bar{\mathbf{x}}'_j$, a private sanitizing key k_{j-1}, the sum of all public signing key, the new private sanitization key k_j, and outputs the sanitizable result R_j.

$$R_j \leftarrow SubSanit(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}'_j, \sum_{i=1}^{M} S_i, k_j, k_{j-1})$$

- **Verify**: It takes the message $\bar{\mathbf{x}}_j$, the sum of public signing key $\sum_{i=1}^{M} S_i$, a public sanitization key K_j, the random integer R_j and the received signature σ as input, and outputs a bit b. If $b = 1$, the integrity of the processed message can be guaranteed.

$$b \leftarrow Verify(\bar{\mathbf{x}}_j, \sum_{i=1}^{M} S_i, K_j, R_j, \sigma_{agg})$$

There are two advantages of ECMSS. First, once each vEN selects the cells, it can remove them from the original message and transmit the rest of the cells to its offspring. The offspring vEN can verify the rest of the cells without knowing the cells selected by its ancestor vEN. Second, through the re-key function, each vEN can substitute the current sanitation key with its own sanitization key. Therefore, if a vEN is not involved in verification, it cannot try to attend the multi-hop verification because it does not know the current sanitization key.

Then, we give a specific introduction with a concrete example.

In the Setup Algorithm, the generation of keys entails the creation of two types: signing and sanitization. Each IoT device has its public and private signing key and the public sanitizing key. Additionally, each vEN has unique public and private sanitizing keys.

The h-th IoT device generates the signature to ensure the message integrity according to the Sign algorithm

$$\sigma_h = H(\mathbf{x}_h)G + r_h K_0 + s_h r_0 G \tag{5.5}$$

In Eq. 5.5, $H(\mathbf{x}_h)G+rK_0$ is based on ECCH in Chapter 2.2.3, and $H(\mathbf{x}_h)G+sr_0G$ can fulfill the signature aggregation with different signing keys according to the aggregate algorithm. In other words, it can achieve subset aggregation.

$$\sigma_{agg} = \sum_{i=1}^{M}\sum_{h=1}^{T_i}(H(\mathbf{x}_h)G + r_h K_0 + s_h r_0 G) \tag{5.6}$$

Following the aggregate algorithm, Eq. 5.6 shows the final aggregated signature, which is used as the signature of all the messages from the IoT nodes. Besides, the gEN computes $H_{agg} = \sum_{i=1}^{M} \sum_{h=1}^{T_i} H(\mathbf{x}_h)$. S is the aggregation of all the public keys. To simplify the description, we define $\mathbf{x}$ as the final message, which concatenates all the messages.

There are two phases in the multi-hop verification: the offloading phase and the feedback phase. In other words, the vEN$_j$ either selects a subset of the cells and transmits the rest of the cells to their offspring vEN, or begins to verify the incoming cells and start the feedback phase.

For the first situation, the vEN$_j$ selects a number of cells in the message and applies the SubSanit Algorithm. In this scheme, it sanitizes them by the attributes of the cells it selects according to Eq. 5.7.

$$R_j = k_j^{-1}(H(\bar{\mathbf{x}}_j) - att(rem(\bar{\mathbf{x}}_j)) + k_{j-1}K_{j-1}) \tag{5.7}$$

After that, the vEN$_j$ transmits the $rem(\bar{\mathbf{x}}_j)$, R_j, $att(rem(\bar{\mathbf{x}}_j))$ and $H_{agg} - H(\bar{\mathbf{x}}_j)$ to its offspring (i.e., vEN$_{j+1}$). Since the vEN$_j$ has selected a subset of the message, it can separate them from the message to reduce the communication cost.

For the second situation, when vEN$_N$ has enough computation ability to verify the incoming message, it checks the $\bar{\mathbf{x}}_N$ integrity.

$$\sigma'_{agg} = (H(\bar{\mathbf{x}}_N) + aux)G + R_N K_0 + \sum_{j=1}^{N-1}(att(rem(\bar{\mathbf{x}}_j)))G + r_0 S \tag{5.8}$$

Inside, $aux = H_{agg} - \sum_{j=1}^{N-1}(H(\bar{\mathbf{x}}_j))$. If vEN$_N$ finishes the verification tasks, it transmits $H(\bar{\mathbf{x}}_N)G$ to its ancestor (vEN$_{N-1}$). Subsequently, the offloading phase stops and the feedback phase begins.

For the rest $N-1$ vENs, after receiving $H(\bar{\mathbf{x}}_{j+1})G$, it only needs to verify $\bar{\mathbf{x}}_j$ according to Eq. 5.9.

$$\sigma'_{agg} = (H(\bar{\mathbf{x}}_j))G + R_j K_0 + \sum_{l=1}^{j-1}(att(rem(\bar{\mathbf{x}}_l)))G + r_0 S \tag{5.9}$$

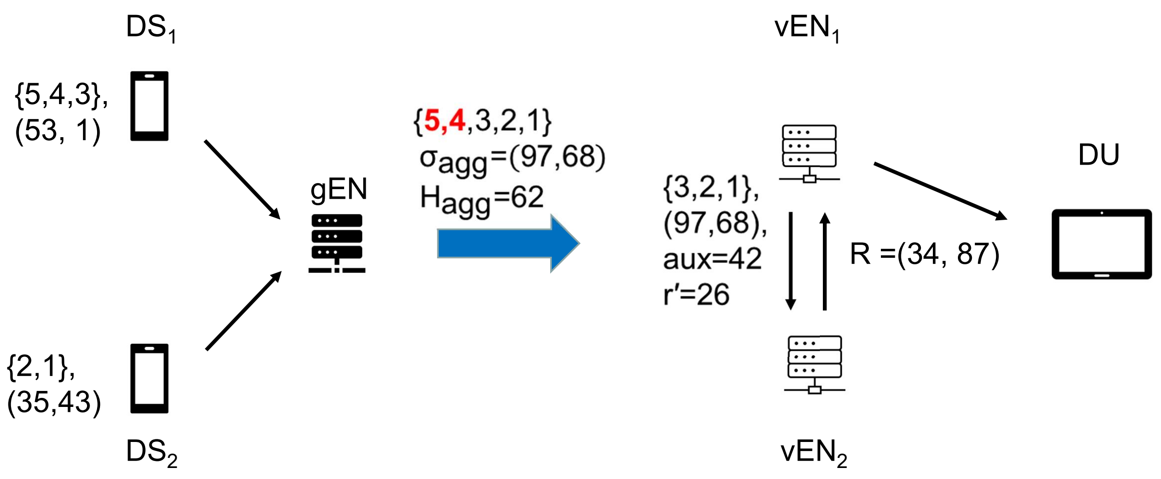


Figure 5.3: The Running Example of ECMSS with Two vENs and One gEN

Then the vEN_{j} computes and sends $H(\bar{\mathbf{x}}_j)G$ to its ancestor.

The signature can also support one-hop verification when one vEN has enough computation ability. In this situation, it does not need to sanitize any subset of the message. Based on Eq. 5.10, the vEN can verify all the cells.

$$\sigma'_{agg} = \sum_{i=1}^{M}\sum_{h=1}^{T_i}(H(\mathbf{x}_h)G) + R_0k_0 + r_0S \tag{5.10}$$

We use a generic example to show how it works.

Example 7. *Figure 5.3 demonstrates an example of ECMSS with one gEN and two vENs. The elliptic curve is $E_{101}(1,1)$, which is $y^2 = x^3 + x + 1 \bmod 101$. DS_1 and DS_2 own the message $(5,4,3)$ and $(2,1)$. According to their signing keys ($s_1 = 3$ and $s_2 = 2$), random number ($r_0 = 1$, $r_1 = 2$ and $r_2 = 3$), and public sanitization key ($K_0 = (4,66)$), they generate two signatures ($\sigma_1 = (53,1)$ and $\sigma_2 = (35,43)$). The gEN concatenates the incoming message ($(5,4,3,2,1)$), and aggregate the signature ($(97,68)$) and hash result ($H_{agg} = 62$).*

On the verifier side, the first vEN can choose the first two cells for verification. It sanitizes the subset $(5,4)$ according to the secret sanitization key ($k_0 = 10$), and

Table 5.3: Variables List for Energy Optimization

Parameter	**Instruction**	**Unit**
i, N	Index, the number of vEN	n/a
$\|\mathbf{x}\|$	The number of cells in the message	n/a
E_i^{max}	The upper-bound energy of the i-th edge node	J
W_i^{max}	The upper-bound workload of the i-th edge node	bit
B	The cell size of the message	bit
τ_i	The number of verified cells by the i-th edge node	n/a
μ	The coefficient related to the chip architecture	n/a
f_i	The CPU frequency for the i-th edge node	GHz
β	Required CPU cycles to compute 1-bit of input data	cycles/bit

computes $r' = 26$ and $aux = 42$. vEN$_2$ checks the integrity of the rest cells (i.e., $(3, 2, 1)$) according to Eq. 5.9. If their integrity is guaranteed, the vEN computes $(34, 87)$, which is relevant to verified cells and aux, and transmits it to vEN$_1$. For vEN$_1$, when it receives the computing result, it checks the integrity of the rest cells.

The example shows that the previous $N - 1$ vENs participate in offloading and feedback phases. On the contrary, the last vEN only does the feedback phase. The more cells it verifies, the less time the system consumes.

5.3 Multi-hop Verification Optimization

In this section, we discuss the optimization problem in the multi-hop verification. Table 5.3 shows the notations in optimization. Since each vEN can autonomously select the cells for verification. Optimization is critical. Our objective is to minimize energy consumption for all the vENs. There are some limitations.

1. The number of cells verified by each vEN cannot exceed its upper-bound work-

load.

2. The energy consumption for verifying cannot exceed its upper-bound energy.
3. The vENs can finish the verification tasks.

The energy consumption in the system contains two parts: energy for message execution (E^{verify}) and energy for message transmission (E^{tran}). The energy model follows like that in [47][95].

For the i-th vEN, the energy consumption is computed according to Eq. 5.11.

$$E_i^{\text{verify}} = \mu\beta f_i^2 B\tau_i \tag{5.11}$$

Concurrently, E^{tran} is computed according to Eq. 5.12.

$$E_i^{\text{tran}} = \frac{P(|\mathbf{x}| - \tau_i)}{T_R B_w} \tag{5.12}$$

Inside, P is the maximum energy for transmission, B_w is the transmission bandwidth, T_R is the transmission rate of the system, which can be computed according to Eq. 5.13.

$$T_R = \log_2{(1 + \frac{PC_g}{\sigma_n^2})} \tag{5.13}$$

σ_n^2 denotes noise power and C_g is the channel gain.

Besides, the workload of each edge node is as follows,

$$W_i = B\tau_i \tag{5.14}$$

According to the limitation, we can derive that

$$\min \quad \sum_{i=1}^{N} E_i^{\text{verify}} + \sum_{i=1}^{N} E_i^{\text{tran}} \tag{5.15a}$$

$$\textbf{s.t.} \quad \forall i, 0 \leq \mu\beta f_i^2 B\tau_i \leq E_i^{max} \tag{5.15b}$$

$$\forall i, 0 \leq B\tau_i \leq W_i^{max} \tag{5.15c}$$

$$\sum_{i=1}^{N} \tau_i = |\mathbf{x}| \tag{5.15d}$$

$$\tau_i \text{ is integer} \tag{5.15e}$$

Since μ, f_i, β, and the parameters of E_{tran} have been determined by the hardware of the edge node, the optimization can be easily solved. We can discover that Eq. 5.15b, Eq. 5.15c, and Eq. 5.15d are linear equations, while Eq. 5.15e shows that the optimization is an integer-based optimization problem [90]. Consequently, we can apply mixed-integer programming, which is the mathematical framework for optimizing energy systems.

5.4 Security Analysis

In this section, we show the theoretical security analysis, which includes correctness, unforgeability, and non-transferability.

5.4.1 Correctness

Suppose there are M gENs and N vENs in the network. The final signature can be generated according to Eq. 5.5 and Eq. 5.6. Globally, each gEN will generate $\mathbf{x}_i$, and each vEN will verify $\mathbf{y}_i$. Consequently, we have

$$\sum_{i=1}^{M} H(\mathbf{x}_i) = \sum_{j=1}^{N} H(\mathbf{y}_j) + aux \tag{5.16}$$

The ECCH can be helpful in one-hop situations as a traditional chameleon hashing function. We should promote it to multi-hop situations.

Theorem 9. *Our proposed ECMSS is correct for multi-hop verification.*

Proof. According to Eq. 5.5, the sr_0G will not be sanitized during the transmission. Therefore, we define the state as the result of ECCH. Similarly, vEN$_2$ to vEN$_{N-1}$, they will choose a subset of the message and sanitize the state.

$$\begin{aligned}
state &= (\sum_{j=1}^{N}(H(\mathbf{y}_j)) + aux)G + R_0K_0 \\
&= (\sum_{j=2}^{N}(H(\mathbf{y}_j)) + aux)G + H(\mathbf{y}_1)G + R_0K_0 \\
&= (\sum_{j=2}^{N}(H(\mathbf{y}_j)) + aux)G + H(att(||_{j=2}^{N}(\mathbf{y}_j)))G + R_1K_1 \\
&= (\sum_{j=3}^{N}(H(\mathbf{y}_j)) + aux)G + H(\mathbf{y}_2)G + H(att(||_{j=2}^{N}(\mathbf{y}_j)))G + R_1K_1 \\
&= (\sum_{j=3}^{N}(H(\mathbf{y}_j)) + aux)G + H(att(||_{j=3}^{N}(\mathbf{y}_j)))G + H(att(||_{j=2}^{N}(\mathbf{y}_j)))G + R_2K_2 \\
&= \dots \\
&= (H(\mathbf{y}_N)) + aux)G + \sum_{l=2}^{N-1} H(att(||_{j=l}^{N}(\mathbf{y}_j))) + R_NK_N
\end{aligned}$$

□

5.4.2 Unforgeability

The verification is done by the vEN, and the aggregation is finished before the verification. We define the unforgeability only based on one public key as follows,

Definition 7. ***(Unforgeability)***. *ECMSS(Setup, SanKeyGen, Sign, Verify, SubSanit) is unforgeable under adaptive chosen attacks if for any efficient algorithm $\mathcal{A}$ that the experiment Unforgeability$_{\mathcal{A}}^{ECMSS}$ evaluates to 1 is negligible.*

Experiment $Unforgeability_{\mathcal{A}}^{ECMSS}$
-$(pk_{sign}, sk_{sign}) \leftarrow Setup(1^{\lambda})$
-$(pk_{sanit}, sk_{sanit}) \leftarrow SanKeyGen(1^{\lambda})$
-$\sigma \leftarrow \mathcal{A}^{Sign(sk_{sign},\dot{})}(pk_{sign})$
-*for* $i = 1, 2, ..., q$, *denoted by* σ_i, *the queries to the oracle*($\mathcal{O}$) *Sign return 1 iff Verify*$(pk_{sign}, pk_{sanit}, \sigma) = 1$

Specifically, the adversary can run a probabilistic algorithm in at most t steps and make no more than q_a queries to the Sign algorithm and no more than q_b queries to the SubSanit algorithm to forge the ECMSS with the probability of success smaller than ϵ on problem instances of size k. Therefore, we have the theorem that.

Theorem 10. *The ECMSS is* $(\epsilon, k, q_a, q_b, t)-$*unforgeable in the random oracle model.*

Proof. There are two situations in which the adversary can find efficient solutions to forge the ECMSS. First, he can easily forge signature schemes. Second, he can discover the collision of the chameleon hash without knowing the sanitization key.

We say that ECMSS is $(\epsilon, k, q_a, q_b, t)-$ unforgeable. Specifically, the adversary can run a probabilistic algorithm in at most t steps and make no more than q_a queries to the Sign algorithm and no more than q_b queries to the SubSanit algorithm to forge the ECMSS with the probability of success smaller than ϵ on problem instances of size k.

If the $\mathcal{A}$ be an $(\epsilon, k, q_a, q_b, t)-$forger, the two situations can be shown as follows,

1. There exists an $(\epsilon', k, q_a, t')-$forger of the ECMSS scheme.

2. There exists an $(\epsilon'', k, q_b, t'')-$forger of the ECCH.

The quantities are as follows,

$$\epsilon \leq \epsilon' + \epsilon'' \tag{5.18}$$

$$t \geq t' - q_b t_{\text{collision}} \tag{5.19}$$

$$t \geq t'' - q_a t_{\text{sign}} \tag{5.20}$$

Insides, $t_{\text{collision}}$ and t_{sign} are the maximum running times of collision finding and signing forging algorithms on instances of size k.

We define S as the intermediate value so that $\sigma = Sign_{sk_{sign}}(S)$. Once $\mathcal{A}$ succeeds in computing a signature σ on a new message $\mathbf{x}$, only two cases will happen.

Case 1 Every query to the oracle $\mathcal{O}^{sk_{sign}}$ during $\mathcal{A}$'s execution resulted in $\sigma' = Sign_{sk_{sign}}(S')$, which will be different to $\sigma = Sign_{sk_{sign}}(S)$.

Case 2 There is a query $\mathbf{x}_i$ to the $\mathcal{O}^{sk_{sign}}$ so that the response σ_i equals to $Sign_{sk_{sign}}(S)$, with $\mathbf{x}_i \neq \mathbf{x}$

For the first case, suppose another adversary $\mathcal{B}$ can generate public and private sanitization keys (sk_{sanit}, pk_{sanit}). He uses sk_{sanit} to create the chameleon hash and send pk_{sanit} to $\mathcal{A}$. To answer the signature queries from $\mathcal{A}$, $\mathcal{B}$ resorts to its signing oracle for the underlying signature schemes. After $\mathcal{A}$ generates the signature, S will be outputted and recorded.

After that, $\mathcal{B}$ reads σ from the record and terminates when $\mathcal{A}$ successfully computes the same σ. The execution time is $t' = t + q_b t_{\text{collision}}$, where t is the number of steps used by $\mathcal{A}$. Therefore, Eq. 5.19 is achieved.

For the second case, a new adversary $\mathcal{C}$ is built for the chameleon hash algorithm. In the beginning, $\mathcal{C}$ generates signing key pairs (sk_{sanit}, pk_{sanit}). He uses sk_{sign} with the underlying signing algorithm $sign(\dot{)}$ to emulate $\mathcal{O}^{sk_{sign}}$ and send pk_{sign} to $\mathcal{A}$. To answer the sanitization queries, $\mathcal{C}$ resorts to the collision-finding oracle for the

ECCH. Once $\mathcal{A}$ computed σ, $\mathcal{C}$ retrieves S and compares it with S_i that recorded in $\mathcal{A}$. Consequently, there is at least one query that differs from $\mathbf{x}$ but such that S_i is equal to S. Therefore, we can derive that,

$$\text{state} = ECCH(\mathbf{x}, pk_{sanit}, r) = ECCH(\mathbf{x}_i, pk_{sanit}, r_i)$$

Then $\mathcal{C}$ outputs state to seek collision against $\mathcal{A}$. According to the $\mathbf{x}, r$, $\mathcal{C}$ can succeed with the execution time $t'' = t + q_a t_{sign}$. Therefore, Eq. 5.20 is achieved.

□

5.4.3 Non-transferability

Definition 8. ***(Non-transferability)****: A signature issued to a designated recipient cannot be validated by another party.*

Theorem 11. *The ECMSS has the property of non-transferability for multi-hop verification.*

Proof. Based on the semantic security in Chapter 2.2.3, the traditional chameleon hash-based signature has the property of non-transferability.

In the multi-hop scenario, suppose the undesignated party $\mathcal{A}$ try to verify the message instead of vEN$_j$ based on Eq. 5.9, he just reads σ, pk_{sign} and pk^0_{sanit}. According to Eq. 5.7, the vEN$_{j-1}$ has sanitized the subset of the message. In the meantime, he changes the sanitization key. $\mathcal{D}$ cannot verify the message in this situation. □

5.5 Experiments

In this section, we use the experiments to show the performance of ECMSS and compare it with the threshold signature [91], Identity-based Sanitizable Signature

Table 5.4: Variables List for Network Setting in The Experiment

Parameter	B_w	C_g	σ_n^2	β	μ	P
Instruction	1MHz	2.585	-100 dBm	10^3 cycles/bit	10^{-27}	30 dBm

(IDSS) [71] and Schnorr signature [21]. The experiments will be partitioned into three components. First, we will show the performance of ECMSS. Second, we will compare the performance of ECMSS, IDSS, Schnorr signature, and threshold signature (T-ECDSA). Finally, we will show the optimization experiments of ECMSS. We make the comparison between two cell selection strategies. The E_{avg} means that each vEN will choose the same number of cells for verification. The E_{opt} means that each vEN will choose the number of cells equal to the optimization result. The message is from the practical dataset. Each message contains 30000 cells, and the cell size is 20 bits. The network setting follows Table 5.4. The CPU frequency and the computation ability are distributed in $[1\text{GHz}, 20\text{GHz}]$ and [1FLOPS, 20FLOPS].

In ECMSS, IDSS, and T-ECDSA schemes, the signing key and nonce are set at a length of 160 bits. Concurrently, the sanitization key also adheres to 160-bit length requirement. The secp256k1 curve is selected as the designated curve. The Schnorr Signature needs 3072 bits prime to reach the same security level. We use virtual machines to simulate the vEN with different computation abilities. The number of CPU is 1, and the RAM is 4GB. The code of each edge node is implemented in Python 3.7. We adopt Floating-point operations per second (FLOPS) to measure the computation ability of the edge node.

We use two datasets. One is the synthetic dataset. The cells in the message are randomly generated with the same cell size. The other is the practical dataset. We use the "Pune Smart City Dataset" from Kaggle. It contains different kinds of smart city data, such as temperature, sound, and light. The data is used to predict the air quality index. The cell size is 20 bit.

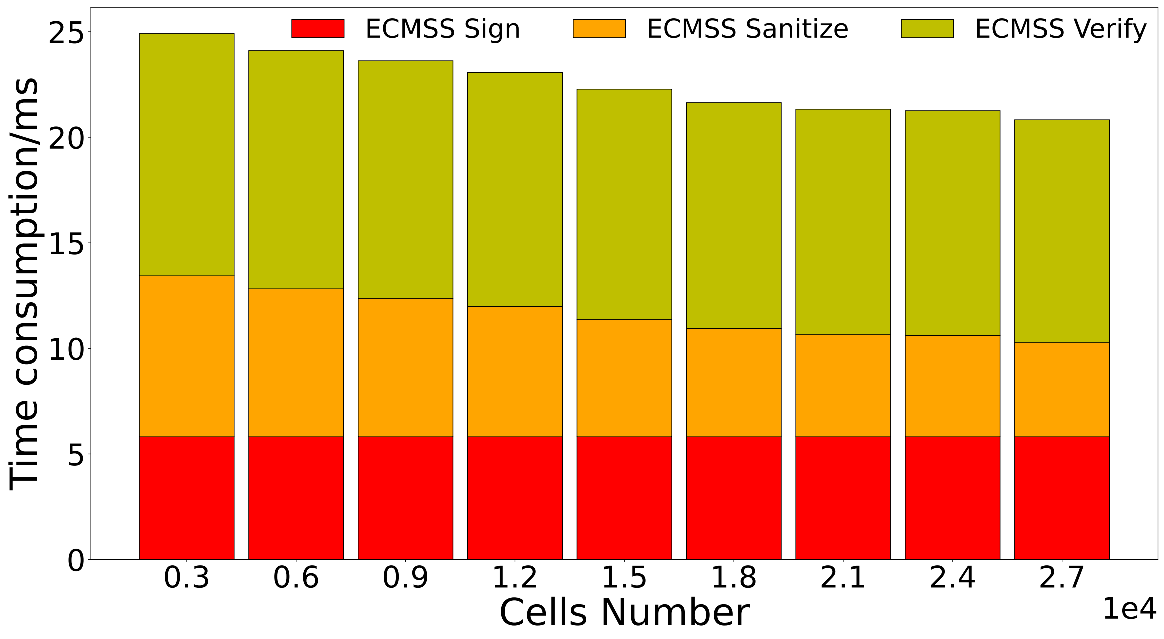


Figure 5.4: The Number of Verified Cells by The Last vEN

5.5.1 The Time Consumption under The Number of Cells Verified by The last vEN

According to Chapter 5.2.2, the last vEN only performs verification tasks, while the other vEN will do both sanitization and verification. We apply two vENs with 10FLOPS and 20FLOPS computation ability. Notably, the final vEN possesses more extraordinary computational ability compared to the previous vEN. There are 30000 cells in one message. The number of cells verified by the last vEN increases from 3000 to 27000.

According to Figure 5.4, the time consumption decreases when the number of cells verified by the last vEN increases. Based on the previous result, once the vEN with the most incredible computation ability is deployed in the last order, the time consumption of the whole system reduces.

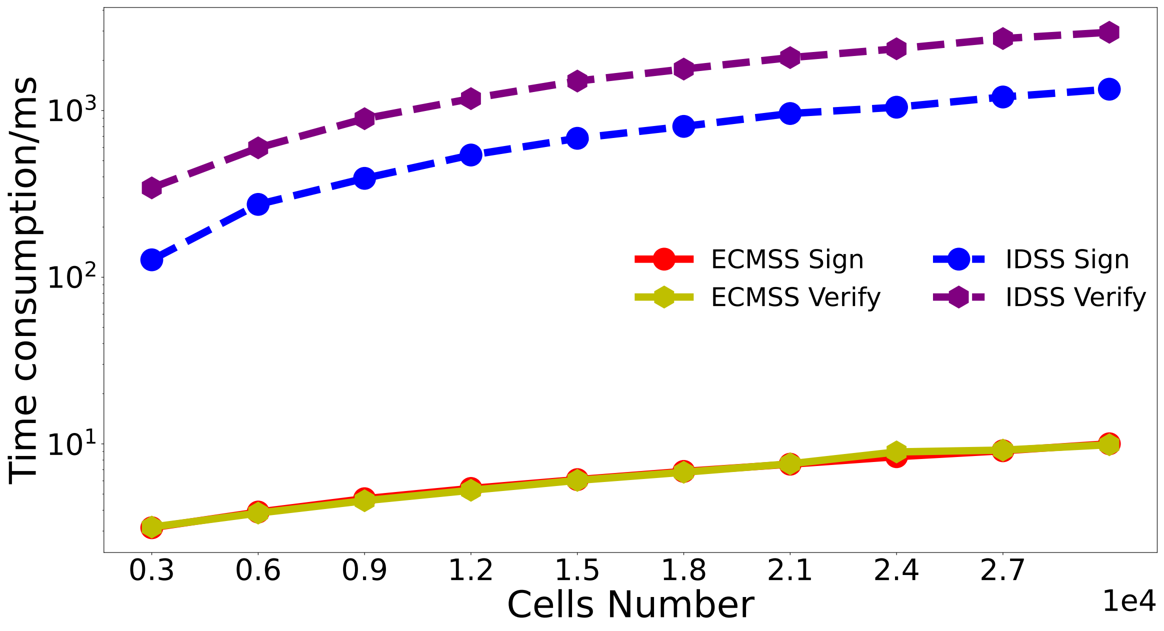


Figure 5.5: The Time Consumption under The Number of Cells

5.5.2 Time Consumption under The Number of Cells

In this experiment, we demonstrate the time consumption of ECMSS and IDSS in one vEN.

The number of cells in one message increases from 3000 to 30000, and the vEN verifies all of them. According to Figure 5.5.2, The time consumption of ECMSS in signing and verifying is smaller than that in IDSS. Moreover, the growth rate of IDSS is monopoly increasing. When the number of cells is 3000, the time consumption of signing and verifying in IDSS is about 40 and 108 times larger than those in ECMSS. Once the number of cells is 30000, the time consumption of signing and verifying in IDSS is about 134 and 298 times larger than those in ECMSS. The reason is that IDSS is based on cells. The signature generation and verification should be done for each cell. Therefore, the time consumption is significant.

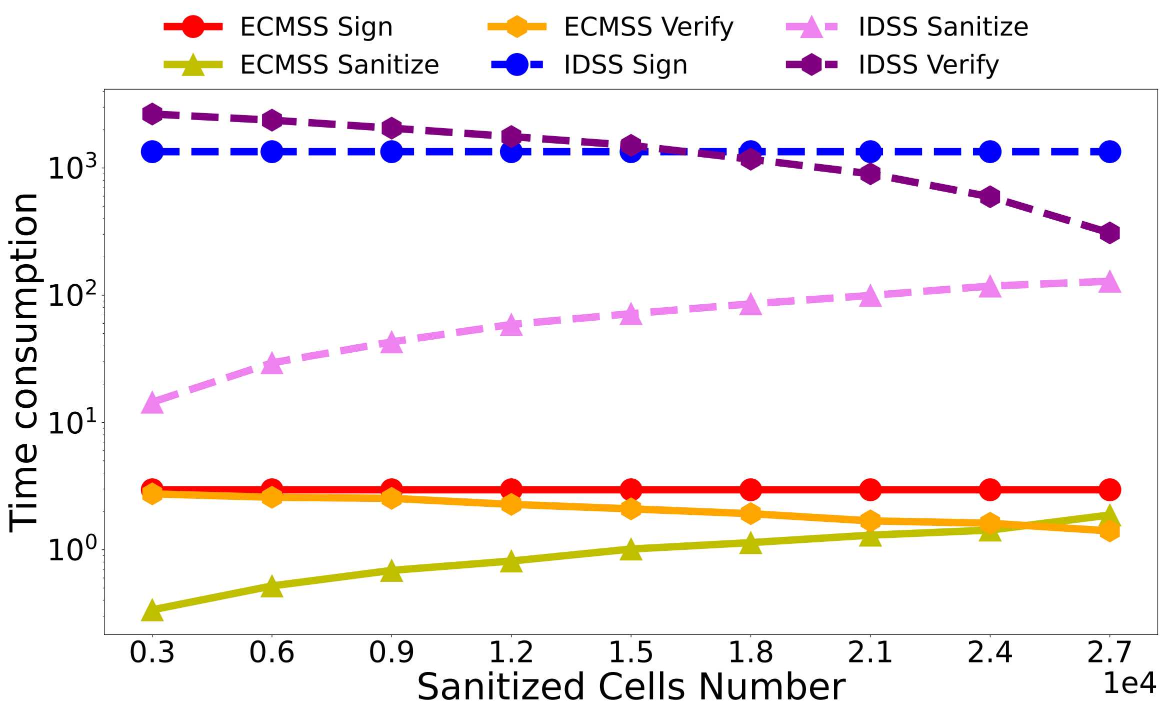


Figure 5.6: The Time Consumption under The Number of Sanitized Cells

5.5.3 Time Consumption under The Number of Sanitized Cells

In this experiment, we compare the performance of ECMSS and IDSS when the number of sanitized cells changes. The number of sanitized cells increases from 3000 to 27000. The participated vEN has the same computation ability.

Since both schemes support subset verification, the time for signing does not change. According to the definition of sanitization, when the number of sanitized cells increases, the number of verified cells reduces. As Figure 5.6 shows, the time for ECMSS and IDSS sanitizing increases due to the same reason. Concurrently, the time consumption of ECMSS and IDSS verifying decreases.

5.5.4 The Number of gEN

In this experiment, we compare the three schemes for subset aggregation. The construction of the Schnorr signature can naturally support the subset aggregation. We assume that the gEN has the same computation ability. There are 100 IoT devices, while each IoT devices has one message with 300 cells and the number of gEN increases from 1 to 5. In IDSS scheme, each IoT device has the same key so that the gEN can aggregate them. However, in ECMSS and Schnorr signature, the key for signature generation is different.

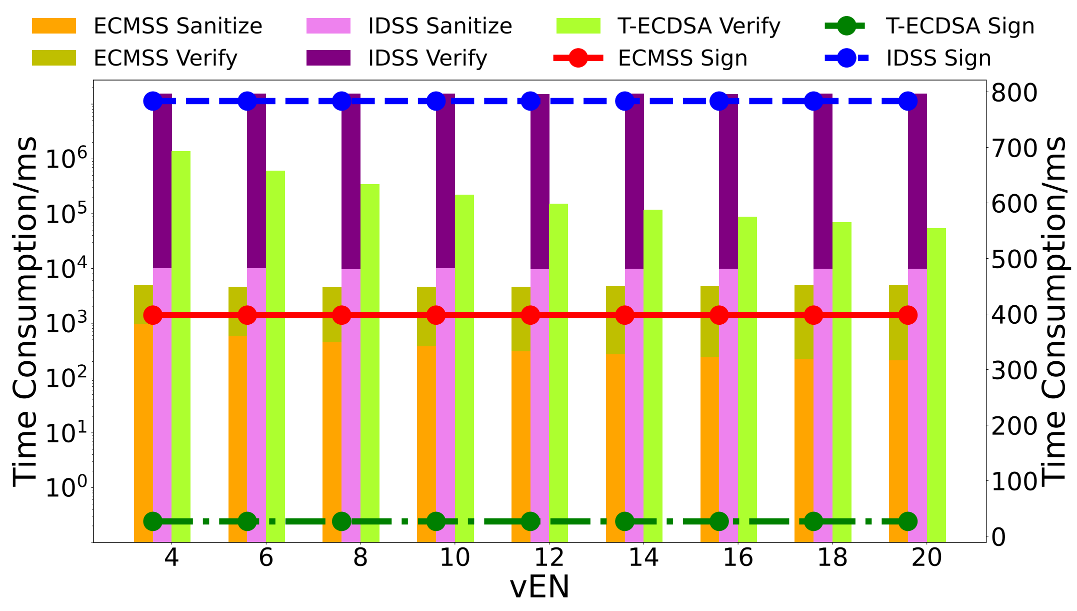
Table 5.5 shows that the number of gEN has no effect on the time for aggregation since the number of signatures for aggregation does not change. The IDSS scheme consumes the most considerable time for aggregation because it divides the messages into several cells and signs them separately. Therefore, the number of signatures is more significant than that of the other two signature schemes, and its time consumption is substantial. To reach the same security level, the Schnorr Signature needs a 3072 bits prime. Consequently, the time consumption becomes enormous.

Table 5.5: Time Consumption of Aggregation Under The Different Number of gEN

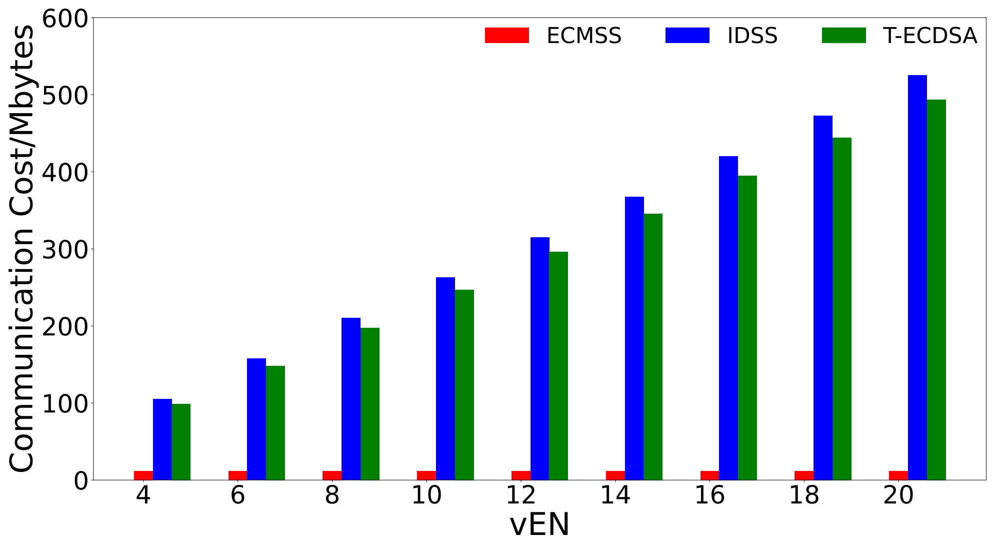
The Number of gEN	1	2	3	4	5
ECMSS	0.375	0.503	0.413	0.421	0.377
IDSS	7588.351	7652.625	7682.882	7716.708	7718.956
Schnorr Signature	9.937	10.987	10.333	9.931	10.331

5.5.5 The Number of vEN

In this experiment, we compare the three schemes in multi-hop verification offloading scenarios. We assume that the computation ability of vEN$_i$ is iFLOPS (e.g., the



(a) Semi-log Plot of The Time Consumption under The Number of vEN



(b) The Communication Cost under The Number of vEN

Figure 5.7: Performance of ECMSS, IDSS, and T-ECDSA

computation ability of vEN_1 is 1FLOP). With the increasing number of vENs, those vENs with strong computation ability will participate. For instance, when the number of vEN is 4, only the vENs with 1, 2, 3, and 4FLOPS computation ability are involved. When the number of vEN is 20, the vEN with 20FLOPS computation ability will participate. We apply the practical dataset about the smart city in this experiment. The threshold in the T-ECDSA scheme is 300, which is 1% of the number of cells. Due to the advanced protocol in transmission (e.g., 5G and WiFi6), the time consumption between two edge nodes can be minimized so that we ignore the time for transmission. The vENs only sanitize and verify the message in ECMSS, and only verify the message in T-ECDSA.

Figure 5.7(a) illustrates the time consumption of each scheme when the number of vEN increases. Notably, the number of vENs does not impact the time consumption of signing in three schemes since there is only one signature for each. For the verifier side, the IDSS consumes the most significant time among the three schemes. The reason has been explained in Chapter 5.5.2. The T-ECDSA scheme should use at least 300 cells to recover the predetermined secret. Thus, signature verification takes a long time. When the number of vEN increases, the sanitizing time decreases, as shown in Chapter 5.5.1.

Figure 5.7(b) demonstrates the communication cost of ECMSS, IDSS, and T-ECDSA among the vENs. The communication cost in IDSS and T-ECDSA is higher than that in ECMSS. On one hand, there are n signatures in IDSS, while only one signature is in ECMSS. On the other hand, although there is only one signature in the T-ECDSA scheme, there are multiple shadows being transmitted with the messages. Consequently, our proposed ECMSS scheme performs well in multi-hop verification scenarios.

In summary, although ECMSS does not perform well on message verification on one node, it performs better in multi-hop verification, which is suitable in collaborative edge computing scenarios in the smart city.

5.5.6 Workload Allocation under The Upper-bound Workload

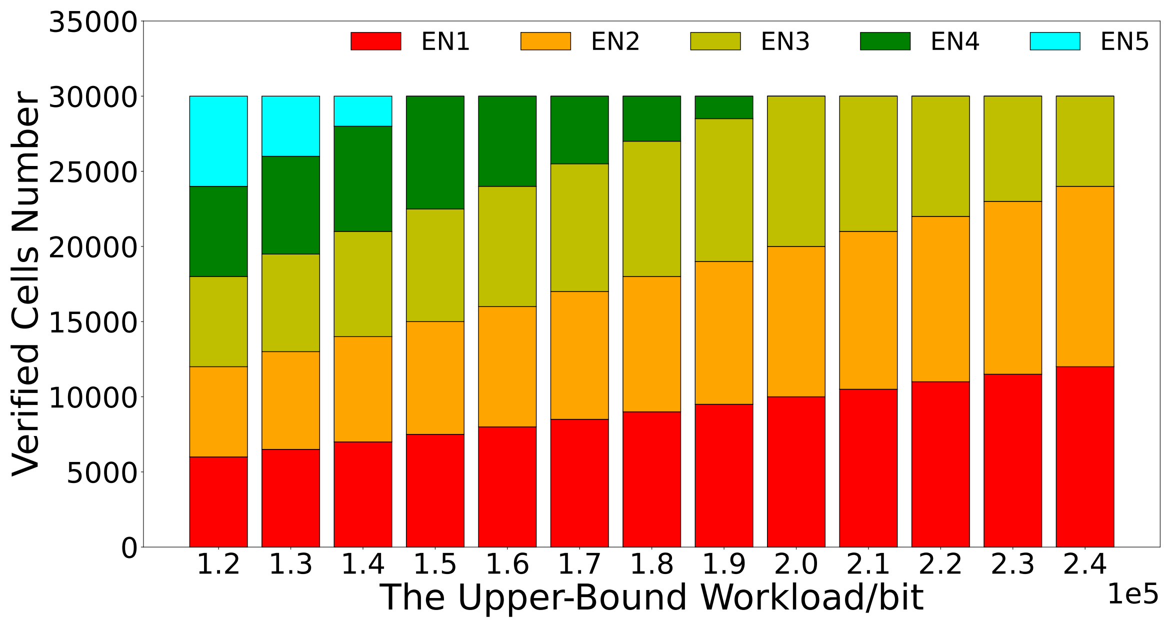
In this experiment, we show how the workload affects the optimization. Five vENs participate in the multi-hop verification offloading. Their CPU frequency increases from 2GHz to 6GHz.

According to Eq. 5.11, the node with high CPU frequency will not be involved in verification to reduce energy consumption. Furthermore, since the workload of each vEN becomes more prominent, we can apply fewer edge nodes to achieve multi-hop verification. Figure 5.8(b) proves that the energy consumption in E_{avg} is larger than in E_{opt}. Therefore, energy optimization is practical.

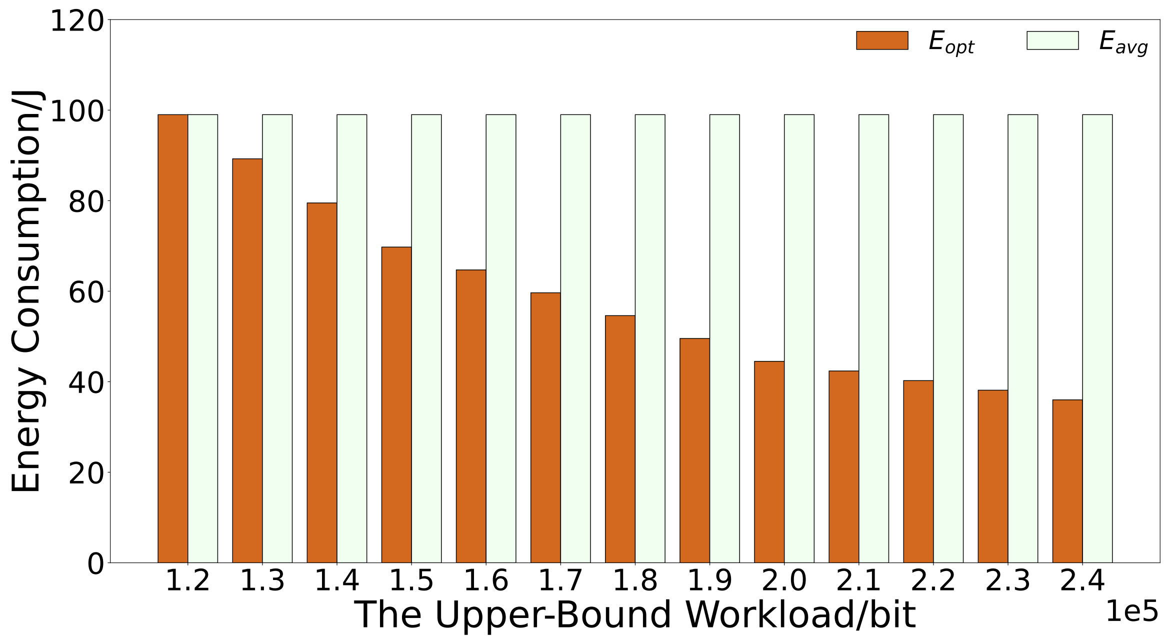
5.5.7 Workload Allocation under The Number of vEN

In this experiment, we show how the number of vEN affects energy consumption and how to optimize the energy consumption in the system. The number of vEN increases from 3 to 7. The CPU frequency of each vEN increases from 2GHz to 8GHz. For instance, when the number of vENs is 4, only the vENs with 2, 3, 4, and 5GHz CPU frequency are involved. To ensure that all vENs participate in verification, we consider that each vEN verifies at least 1000 cells. The upper-bound workload is stable, which is 20000 bits.

Based on Figure 5.9(a), vEN can randomly choose the cells according to the computation ability of each vEN. Moreover, when the number of vENs is extensive, and their workload is large enough to verify the message, the vENs with significant CPU frequency decrease in order to verify the number of cells. However, since the number of transmissions becomes large, the energy consumption for the whole system increases according to 5.9(b). Moreover, the increase rate is low in the system with the E_{opt}.

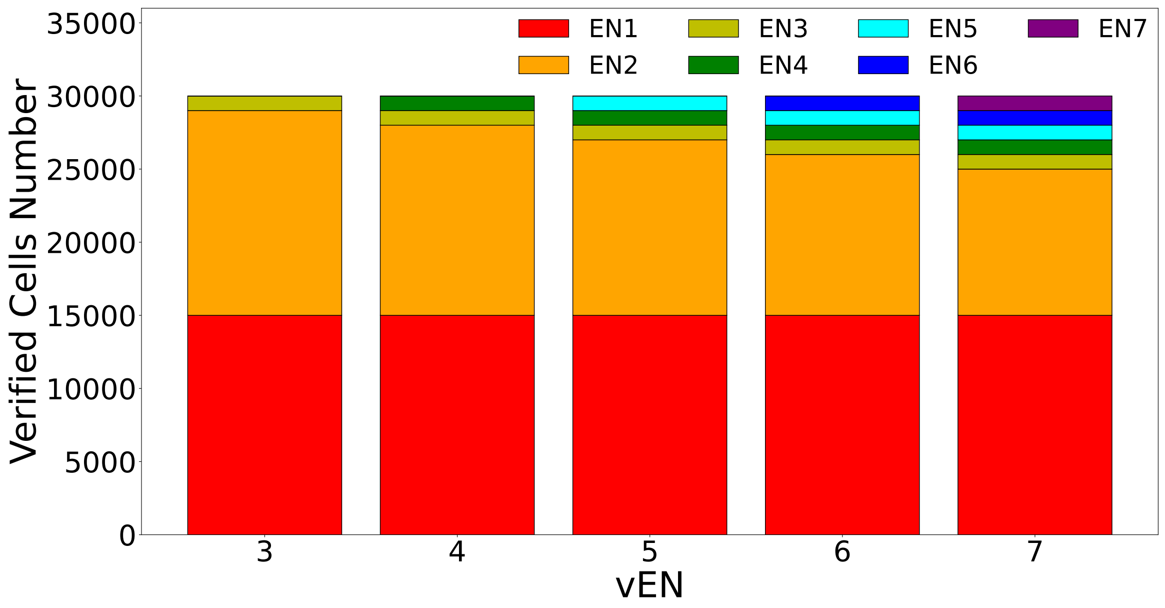


(a) The Number of cells Verified by vEN under The Upper-bound Workload

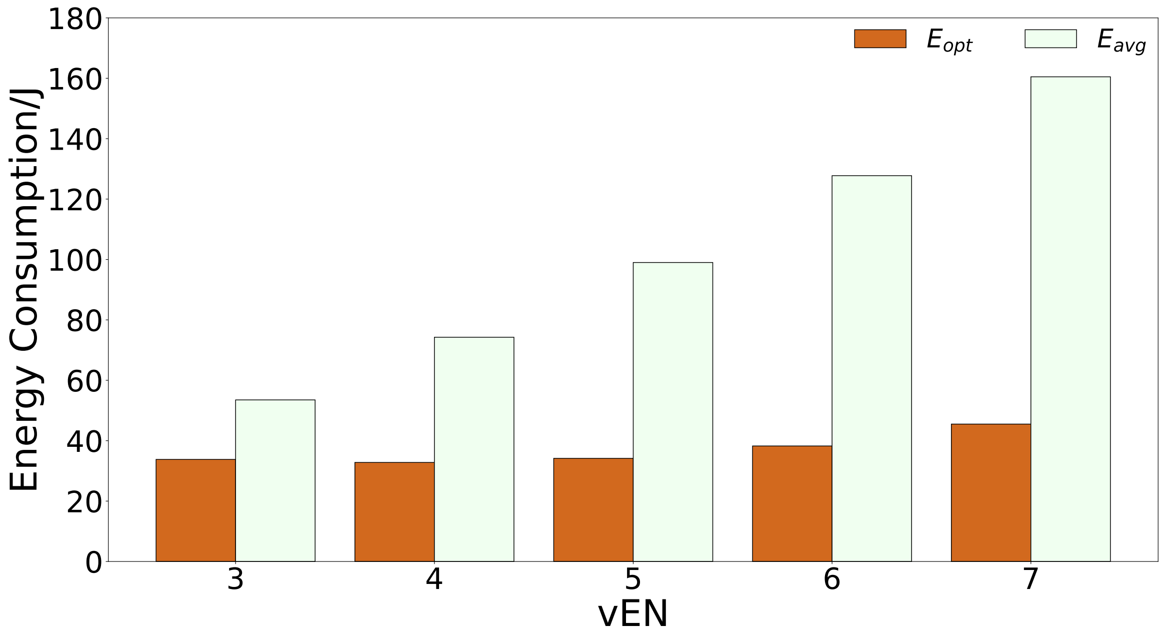


(b) The Energy Consumption under The Upper-bound Workload

Figure 5.8: Performance of Optimization under The Upper-bound Workload



(a) The Number of cells Verified by vEN under The Number of vEN



(b) The Energy Consumption under The Number of vEN

Figure 5.9: Performance of Optimization under The Number of vEN

5.6 Summary

The message integrity problem becomes essential in the edge computing scenario. However, one edge node has a bottleneck in computation ability. Although upgrading hardware can be applied to the edge node to increase its computation ability, we should pay more money to get more workload or computation ability, which is inconvenient. Consequently, subset verification is necessary. In this situation, the edge node can selectively verify a subset of messages and transmit the rest of the messages to resource-sufficient neighbours. In this chapter, we propose the Elliptic Curve Based Multi-hop Sanitizable Signature (ECMSS), which supports subset aggregation and verification in collaborative edge computing scenarios. The theoretical analysis proves the correctness and unforgeability of ECMSS. Furthermore, according to the experiments, ECMSS performs better than the ECDSA scheme in terms of both time consumption and communication cost. According to the performance of ECMSS, it can be applied in some real-life scenarios, such as distributed machine learning and big data analysis. In addition, we provide an energy optimization solution for the system.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

Message integrity is a critical aspect in the IoT scenario. However, one of the challenges we face is the computation ability bottleneck in IoT devices. Although advanced hardware can be applied to increase its computation ability, we should pay more money to get more workload or computation ability, which is inconvenient. This paper aims to shed light on this issue from three different perspectives.

1. **Subset privacy in subset integrity**: As discussed in our proposal, subset privacy is a by-product of the subset integrity problem. Most existing works cannot solve the problem, especially when the sender does not know the privacy request before the message is transmitted. The SPMAC scheme can solve the problem even if the requested cells are randomly deployed in the message. In other words, this scheme is suitable in high-dimension messages, which have been widely used in machine learning scenarios. Compared with the PMAC scheme, it can reach the same security with higher efficiency.

2. **Subset integrity between two entities**: The subset integrity between two

entities is a particular situation. When one of the entities has more extraordinary computation ability than the other, PV-MAC can be applied in this situation. The transmission delay can be ignored thanks to fog computing, while the computation ability can be guaranteed. Furthermore, we provide a workload optimization solution in this scheme to minimize the energy consumption of the data user.

3. **Subset integrity among multiple entities**: Throughout this study, it is easy to find that the problem is in a general situation. The ECMSS scheme can be used when several spare edge nodes are deployed in the network. This solution has further application in a distributed machine-learning scenario. In addition, we provide an energy optimization solution for the system.

To conclude, message integrity remains a pressing concern in the IoT scenario. When IoT devices confront computation ability limitations, subset integrity becomes essential. With the improvement in fog and edge computing technology, verification tasks can be offloaded to other entities. It is imperative to empower the cooperated entities to selectively and optimally verify the message subset.

6.2 Future Work

In the future, we will devise a novel scheme to fulfill verification and verifiable computation. Presently, most fog and edge computing are provided by a third party, so they cannot be fully trusted. Therefore, our proposed schemes may not be universally applicable. We will study lightweight integrity assurance schemes with a half-trusted third party for partial verification. It is more complex than the system model presented in this report because the data user cannot rely solely on the verification result. Specifically, the data user must verify the results obtained from the fog node without recomputing them. In our second work, the verification works are utterly outsourced to the edge nodes. Thus, the verifiable computation is much more critical.

References

[1] IoT Analytics. Global iot market size to grow 19despite economic downturn. `https://iot-analytics.com/iot-market-size/`, February 7, 2023.

[2] Julio Arauz and Tony Fynn-Cudjoe. Actuator quality in the internet of things. In *2013 IEEE International Conference on Sensing, Communications and Networking (SECON)*, pages 34–42. IEEE, 2013.

[3] Muhammad Arif, Guojun Wang, and Valentina Emilia Balas. Secure vanets: trusted communication scheme between vehicles and infrastructure based on fog computing. *Stud. Inform. Control*, 27(2):235–246, 2018.

[4] Frederik Armknecht, Paul Walther, Gene Tsudik, Martin Beck, and Thorsten Strufe. Promacs: Progressive and resynchronizing macs for continuous efficient authentication of message streams. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 211–223, 2020.

[5] Kondapally Ashritha, M Sindhu, and K. V. Lakshmy. Redactable blockchain using enhanced chameleon hash function. In *International Conference on Advanced Computing and Communication Systems*, 2019.

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

[7] Giuseppe Ateniese, Daniel H Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable signatures. In *European Symposium on Research in Computer Security*, pages 159–177. Springer, 2005.

[8] Mohsen Bafandehkar, Sharifah Md Yasin, Ramlan Mahmod, and Zurina Mohd Hanapi. Comparison of ecc and rsa algorithm in resource constrained devices. In *2013 International Conference on IT Convergence and Security (ICITCS)*, pages 1–3. IEEE, 2013.

[9] Luiz André Barroso. The price of performance: An economic case for chip multiprocessing. *Queue*, 3(7):48–53, 2005.

[10] Padma Bh, D Chandravathi, and P Prapoorna Roja. Encoding and decoding of a message in the implementation of elliptic curve cryptography using koblitz's method. *International Journal on Computer Science and Engineering*, 2(5):1904–1907, 2010.

[11] Estuardo Alpirez Bock, Alessandro Amadori, Chris Brzuska, and Wil Michiels. On the security goals of white-box cryptography. *IACR transactions on cryptographic hardware and embedded systems*, pages 327–357, 2020.

[12] Anne Broadbent and Evelyn Wainewright. Efficient simulation for quantum message authentication. In *International Conference on Information Theoretic Security*, pages 72–91. Springer, 2016.

[13] Christina Brzuska, Heike Busch, Oezguer Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, et al. Redactable signatures for tree-structured data: Definitions and constructions. In *Applied Cryptography and Network Security: 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings 8*, pages 87–104. Springer, 2010.

[14] Jan Camenisch, Ueli Maurer, and Markus Stadler. Digital payment systems with passive anonymity-revoking trustees. *Journal of Computer Security*, 5(1):69–89, 1997.

[15] Luis M Candanedo and Véronique Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and Buildings*, 112:28–39, 2016.

[16] Hervé Chabanne, Houssem Maghrebi, and Emmanuel Prouff. Linear repairing codes and side-channel attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 118–141, 2018.

[17] Zheng-Yi Chai, Ying-Jie Zhao, and Ya-Lun Li. Multi-task computation offloading based on evolutionary multi-objective optimization in industrial internet of things. *IEEE Internet of Things Journal*, 2024.

[18] Qi Chen, Chunming Tang, and Zhiqiang Lin. Efficient explicit constructions of multipartite secret sharing schemes. *IEEE Transactions on Information Theory*, 68(1):601–631, 2021.

[19] Wang Chen. An ibe-based security scheme on internet of things. In *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, volume 3, pages 1046–1049. IEEE, 2012.

[20] Xiaofeng Chen, Fangguo Zhang, and Kwangjo Kim. Chameleon hashing without key exposure. In *International Conference on Information Security*, pages 87–98. Springer, 2004.

[21] Yanbo Chen and Yunlei Zhao. Half-aggregation of schnorr signatures with tight reductions. In *Computer Security–ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part II*, pages 385–404. Springer, 2022.

[22] Sherman S. M. Chow. Verifiable pairing and its applications. In *Information Security Applications, International Workshop, Wisa, Jeju Island, Korea, August, Revised Selected Papers*, 2004.

[23] Sherman S. M. Chow, Man Ho Au, and Willy Susilo. Server-aided signatures verification secure against collusion attack. *Information Security Technical Report*, 17(3):46–57, 2013.

[24] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA international conference on cryptography and coding*, pages 360–363. Springer, 2001.

[25] Lizhen Cui, Jing Chen, Wei He, Hui Li, Wei Guo, and Zhiyuan Su. Achieving approximate global optimization of truth inference for crowdsourcing microtasks. *Data Science and Engineering*, 6(3):294–309, 2021.

[26] Pengfei Deng, Shaohua Hong, Jie Qi, Lin Wang, and Haixin Sun. A lightweight transformer-based approach of specific emitter identification for the automatic identification system. *IEEE Transactions on Information Forensics and Security*, 2023.

[27] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H Luan, and Hao Liang. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE internet of things journal*, 3(6):1171–1181, 2016.

[28] Jianqi Du, Fenghao Xu, Chennan Zhang, Zidong Zhang, Xiaoyin Liu, Pengcheng Ren, Wenrui Diao, Shanqing Guo, and Kehuan Zhang. Identifying the ble misconfigurations of iot devices through companion mobile apps. In *2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 343–351. IEEE, 2022.

[29] Jing Du, Jianliang Xu, Xueyan Tang, and Haibo Hu. ipda: supporting privacy-preserving location-based mobile services. In *2007 international conference on mobile data management*, pages 212–214. IEEE, 2007.

[30] Sabyasachi Dutta, Tamal Bhore, M Kutubuddin Sardar, Avishek Adhikari, and Kouichi Sakurai. Visual secret sharing scheme with distributed levels of importance of shadows. In *Proceedings of the Fifth International Conference on Mathematics and Computing: ICMC 2019*, pages 19–32. Springer, 2020.

[31] Amine Erroutbi, Adnane El Hanjri, and Abderrahim Sekkaki. Secure and lightweight hmac mutual authentication protocol for communication between iot devices and fog nodes. In *2019 IEEE International Smart Cities Conference (ISC2)*, pages 251–257. IEEE, 2019.

[32] Xiaoqin Feng, Jianfeng Ma, Huaxiong Wang, Yinbin Miao, Ximeng Liu, and Zhongyuan Jiang. An accessional signature scheme with unmalleable transaction implementation to securely redeem cryptocurrencies. *IEEE Transactions on Information Forensics and Security*, 2023.

[33] Sarah Abdelwahab Gaballah, Christoph Coijanovic, Thorsten Strufe, and Max Mühlhäuser. 2pps—publish/subscribe with provable privacy. In *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, pages 198–209. IEEE, 2021.

[34] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9*, pages 257–273. Springer, 2006.

[35] Oded Goldreich. On expected probabilistic polynomial-time adversaries: A suggestion for restricted definitions and their benefits. *Journal of cryptology*, 23:1–36, 2010.

[36] Louis Goubin and Ange Martinelli. Protecting aes with shamir's secret sharing scheme. In *Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings 13*, pages 79–94. Springer, 2011.

[37] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In *Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part III 37*, pages 66–97. Springer, 2017.

[38] Khizar Hameed, Saurabh Garg, Muhammad Bilal Amin, Byeong Kang, and Abid Khan. A context-aware information-based clone node attack detection scheme in internet of things. *Journal of Network and Computer Applications*, 197:103271, 2022.

[39] Mahmodul Hasan. Real-time and low-cost iot based farming using raspberry pi. *Indonesian Journal of Electrical Engineering and Computer Science*, 17(1):197–204, 2020.

[40] Michael Haus, Muhammad Waqas, Aaron Yi Ding, Yong Li, Sasu Tarkoma, and Jörg Ott. Security and privacy in device-to-device (d2d) communication: A review. *IEEE Communications Surveys & Tutorials*, 19(2):1054–1079, 2017.

[41] Abhishek Hazra, Pradeep Rana, Mainak Adhikari, and Tarachand Amgoth. Fog computing for next-generation internet of things: fundamental, state-of-the-art and research challenges. *Computer Science Review*, 48:100549, 2023.

[42] Xiaofan He, Richeng Jin, and Huaiyu Dai. Multi-hop task offloading with on-the-fly computation for multi-uav remote edge computing. *IEEE Transactions on Communications*, 70(2):1332–1344, 2021.

[43] Verner E Hoggatt Jr and Marjorie Bicknell. Generalized fibonacci polynomials and zeckendorf's theorem. *The Fibonacci Quarterly*, 11(4):399–419, 1973.

[44] Zicong Hong, Wuhui Chen, Huawei Huang, Song Guo, and Zibin Zheng. Multi-hop cooperative computation offloading for industrial iot–edge–cloud computing environments. *IEEE Transactions on Parallel and Distributed Systems*, 30(12):2759–2774, 2019.

[45] Haibo Hu, Qian Chen, and Jianliang Xu. Verdict: Privacy-preserving authentication of range queries in location-based services. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 1312–1315. IEEE, 2013.

[46] Haibo Hu, Qian Chen, Jianliang Xu, and Byron Choi. Assuring spatio-temporal integrity on mobile devices with minimum location disclosure. *IEEE Transactions on Mobile Computing*, 16(11):3000–3013, 2017.

[47] Qiyu Hu, Yunlong Cai, Guanding Yu, Zhijin Qin, Minjian Zhao, and Geoffrey Ye Li. Joint offloading and trajectory design for uav-enabled mobile edge computing systems. *IEEE Internet of Things Journal*, 6(2):1879–1892, 2018.

[48] Haochen Hua, Yutong Li, Tonghe Wang, Nanqing Dong, Wei Li, and Junwei Cao. Edge computing with artificial intelligence: A machine learning perspective. *ACM Computing Surveys*, 55(9):1–35, 2023.

[49] Ke Huang, Xiaosong Zhang, Yi Mu, Xiaofen Wang, Guomin Yang, Xiaojiang Du, Fatemeh Rezaeibagha, Qi Xia, and Mohsen Guizani. Building redactable consortium blockchain for industrial internet-of-things. *IEEE Transactions on Industrial Informatics*, 15(6):3670–3679, 2019.

[50] P. Indyk. Approximate nearest neighbor: Towards removing the curse of dimensionality. In *Proc Symposium on Theory of Computing*, 1998.

[51] Meng Jia, Jing Chen, Kun He, Ruiying Du, Li Zheng, Mingxi Lai, Donghui Wang, and Fei Liu. Redactable blockchain from decentralized chameleon hash functions. *IEEE Transactions on Information Forensics and Security*, 2022.

[52] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.

[53] Robert Johnson, David Molnar, Dawn Song, and David Wagner. Homomorphic signature schemes. In *Cryptographers' track at the RSA conference*, pages 244–262. Springer, 2002.

[54] Rikuhiro Kojima, Dai Yamamoto, Takeshi Shimoyama, Kouichi Yasaki, and Kazuaki Nimura. A new schnorr multi-signatures to support both multiple messages signing and key aggregation. *Journal of Information Processing*, 29:525–536, 2021.

[55] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Hmac: Keyed-hashing for message authentication, 1997.

[56] Hugo Krawczyk and Tal Rabin. Chameleon hashing and signatures. 1998.

[57] Satyam Kumar, Vishnu Asutosh Dasu, Anubhab Baksi, Santanu Sarkar, Dirmanto Jap, Jakub Breier, and Shivam Bhasin. Side channel attack on stream ciphers: A three-step approach to state/key recovery. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 166–191, 2022.

[58] Somansh Kumar and Ashish Jasuja. Air quality monitoring system based on iot using raspberry pi. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 1341–1346. IEEE, 2017.

[59] Asif Ali Laghari, Hui He, Muhammad Shafiq, and Asiya Khan. Assessing effect of cloud distance on end user's quality of experience (qoe). In *2016 2nd IEEE international conference on computer and communications (ICCC)*, pages 500–505. IEEE, 2016.

[60] Russell W. F. Lai, Tao Zhang, Sherman S. M. Chow, and Dominique Schrder. Efficient sanitizable signatures without random oracles. *Springer International Publishing*, 2016.

[61] Russell WF Lai, Raymond KH Tai, Harry WH Wong, and Sherman SM Chow. Multi-key homomorphic signatures unforgeable under insider corruption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 465–492. Springer, 2018.

[62] Guangshun Li, Jiahe Yan, Lu Chen, Junhua Wu, Qingyan Lin, and Ying Zhang. Energy consumption optimization with a delay threshold in cloud-fog cooperation computing. *IEEE access*, 7:159688–159697, 2019.

[63] Song Li, Jinguang Han, Deyu Tong, and Jie Cui. Redactable signature-based public auditing scheme with sensitive data sharing for cloud storage. *IEEE Systems Journal*, 2022.

[64] Tong Li, Wenbin Chen, Yi Tang, and Hongyang Yan. A homomorphic network coding signature scheme for multiple sources and its application in iot. *Security and communication networks*, 2018, 2018.

[65] Yaping Li, Hongyi Yao, Minghua Chen, Sidharth Jaggi, and Alon Rosen. Ripple authentication for network coding. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[66] Qun Lin, Hongyang Yan, Zhengan Huang, Wenbin Chen, Jian Shen, and Yi Tang. An id-based linearly homomorphic signature scheme and its application in blockchain. *IEEE Access*, 6:20632–20640, 2018.

[67] Chang Liu, Rajiv Ranjan, Chi Yang, Xuyun Zhang, Lizhe Wang, and Jinjun Chen. Mur-dpa: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud. *IEEE Transactions on Computers*, 64(9):2609–2622, 2015.

[68] Jianhang Liu, Ning Liu, Lei Liu, Shibao Li, Hailong Zhu, and Peiying Zhang. A proactive stable scheme for vehicular collaborative edge computing. *IEEE Transactions on Vehicular Technology*, 2023.

[69] Lei Liu, Ming Zhao, Miao Yu, Mian Ahmad Jan, Dapeng Lan, and Amirhosein Taherkordi. Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks. *IEEE Transactions on Intelligent Transportation Systems*, 24(2):2169–2182, 2022.

[70] Yujiong Liu, Shangguang Wang, Jie Huang, and Fangchun Yang. A computation offloading algorithm based on game theory for vehicular edge networks. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[71] Zhenpeng Liu, Lele Ren, Ruilin Li, Qiannan Liu, and Yonggang Zhao. Id-based sanitizable signature data integrity auditing scheme with privacy-preserving. *Computers & Security*, 121:102858, 2022.

[72] Chen Lyu, Amit Pande, Xinlei Wang, Jindan Zhu, Dawu Gu, and Prasant Mohapatra. Clip: Continuous location integrity and provenance for mobile phones. In *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, pages 172–180. IEEE, 2015.

[73] Keith M Martin, Josef Pieprzyk, Rei Safavi-Naini, Huaxiong Wang, and Peter R Wild. Threshold macs. In *International Conference on Information Security and Cryptology*, pages 237–252. Springer, 2002.

[74] Mehrdad Nojoumian and Douglas R Stinson. Sequential secret sharing as a new hierarchical access structure. *J. Internet Serv. Inf. Secur.*, 5(2):24–32, 2015.

[75] Goiuri Peralta, Raul G Cid-Fuentes, Josu Bilbao, and Pedro M Crespo. Homomorphic encryption and network coding in iot architectures: Advantages and future challenges. *Electronics*, 8(8):827, 2019.

[76] Pericle Perazzo and Riccardo Xefraj. Smartfly: Fork-free super-light ethereum classic clients for the internet of things. *IEEE Internet of Things Journal*, 2024.

[77] Thomas Peyrin. Improved differential attacks for echo and grøstl. In *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*, pages 370–392. Springer, 2010.

[78] Henrich C Pöhls and Kai Samelin. Accountable redactable signatures. In *2015 10th International Conference on Availability, Reliability and Security*, pages 60–69. IEEE, 2015.

[79] J. Pomykala, Henryk Kuakowski, P. Sapiecha, and Baej Grela. Id-based, proxy, threshold signature scheme. *International Journal of Electronics and Telecommunications*, 2023.

[80] Thomas Rausch, Cosmin Avasalcai, and Schahram Dustdar. Portable energy-aware cluster-based edge computers. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 260–272. IEEE, 2018.

[81] Hassan Rehan. Revolutionizing america's cloud computing the pivotal role of ai in driving innovation and security. *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023*, 2(1):189–208, 2024.

[82] Yuvraj Sahni, Jiannong Cao, Lei Yang, and Yusheng Ji. Multi-hop multi-task partial computation offloading in collaborative edge computing. *IEEE Transactions on Parallel and Distributed Systems*, 32(5):1133–1145, 2020.

[83] Kai Samelin and Daniel Slamanig. Policy-based sanitizable signatures. 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 538–563. Springer, 2020.

[84] Parsa Sarosh, Shabir A Parah, and Ghulam Mohiuddin Bhat. Utilization of secret sharing technology for secure communication: a state-of-the-art review. *Multimedia Tools and Applications*, 80(1):517–541, 2021.

[85] Jayasree Sengupta, Sushmita Ruj, and Sipra Das Bit. A secure fog-based architecture for industrial internet of things and industry 4.0. *IEEE Transactions on Industrial Informatics*, 17(4):2316–2324, 2020.

[86] Nima Seyedtalebi. Algorithms for provisioning edge computing resources to minimize data transport costs. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2019.

[87] Furqan Shahid and Abid Khan. Smart digital signatures (sds): A post-quantum digital signature scheme for distributed ledgers. *Future Generation Computer Systems*, 111:241–253, 2020.

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

[89] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer, 1984.

[90] Mingqian Shao, Yifan Liul, Bin Duo, Jin Ning, Junsong Luo, Xing Zhu, Mingzhe Liu, and Zhengqiang Wang. Joint passive beamforming and elevation angle-dependent trajectory design for ris-aided uav-enabled wireless sensor networks. In *2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 488–496. IEEE, 2022.

[91] Maha Sliti, Mohamed Hamdi, and Noureddine Boudriga. An elliptic threshold signature framework for k-security in wireless sensor networks. In *2008 15th*

IEEE International Conference on Electronics, Circuits and Systems, pages 226–229. IEEE, 2008.

[92] Yuxuan Sun, Xueying Guo, Jinhui Song, Sheng Zhou, Zhiyuan Jiang, Xin Liu, and Zhisheng Niu. Adaptive learning-based task offloading for vehicular edge computing systems. *IEEE Transactions on vehicular technology*, 68(4):3061–3074, 2019.

[93] Manolis Terrovitis and Nikos Mamoulis. Privacy preservation in the publication of trajectories. In *The Ninth International Conference on Mobile Data Management (mdm 2008)*, pages 65–72. IEEE, 2008.

[94] Yao Tian, Xi Zhao, and Xiaofang Zhou. Db-lsh 2.0: Locality-sensitive hashing with query-based dynamic bucketing. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[95] Yan Kyaw Tun, Yu Min Park, Nguyen H Tran, Walid Saad, Shashi Raj Pandey, and Choong Seon Hong. Energy-efficient resource management in uav-assisted mobile edge computing. *IEEE Communications Letters*, 25(1):249–253, 2020.

[96] Tian Wang, Md Zakirul Alam Bhuiyan, Guojun Wang, Lianyong Qi, Jie Wu, and Thaier Hayajneh. Preserving balance between privacy and data integrity in edge-assisted internet of things. *IEEE Internet of Things Journal*, 7(4):2679–2689, 2019.

[97] Yulong Wang, Kun Qin, Yixiang Chen, and Pengxiang Zhao. Detecting anomalous trajectories and behavior patterns using hierarchical clustering from taxi gps data. *ISPRS International Journal of Geo-Information*, 7(1):25, 2018.

[98] Zhiwei Wang, Ruirui Xie, and Shaohui Wang. Attribute-based server-aided verification signature. *IACR Cryptol. ePrint Arch.*, 2013:406, 2014.

[99] Xin Xie, Chentao Wu, Gen Yang, Zongxin Ye, Xubin He, Jie Li, Minyi Guo, Guangtao Xue, Yuanyuan Dong, and Yafei Zhao. Az-recovery: an efficient

crossing-az recovery scheme for erasure coded cloud storage systems. In *2020 International Symposium on Reliable Distributed Systems (SRDS)*, pages 236–245. IEEE, 2020.

[100] Chugui Xu, Ju Ren, Deyu Zhang, and Yaoxue Zhang. Distilling at the edge: A local differential privacy obfuscation framework for iot data analytics. *IEEE Communications Magazine*, 56(8):20–25, 2018.

[101] Zhezhuang Xu, Guanglun Liu, Haotian Yan, Bin Cheng, and Feilong Lin. Trail-based search for efficient event report to mobile actors in wireless sensor and actor networks. *Sensors*, 17(11):2468, 2017.

[102] Andy Yuan Xue, Rui Zhang, Yu Zheng, Xing Xie, Jin Huang, and Zhenghua Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *2013 IEEE 29th international conference on data engineering (ICDE)*, pages 254–265. IEEE, 2013.

[103] Haotian Yan, Haibo Hu, Qingqing Ye, and Li Tang. Spmac: Scalable prefix verifiable message authentication code for internet of things. *IEEE Transactions on Network and Service Management*, 2022.

[104] Haotian Yan, Zhezhuang Xu, Jianfeng He, Liquan Chen, and Hao Jiang. Striped-flooding: Improve scalability and energy efficiency of flooding algorithm in wireless sensor and actor networks. In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2016.

[105] Qingqing Ye, Haibo Hu, Ninghui Li, Xiaofeng Meng, Huadi Zheng, and Haotian Yan. Beyond value perturbation: Local differential privacy in the temporal setting. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[106] Changsheng You, Kaibin Huang, Hyukjin Chae, and Byoung-Hoon Kim. Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, 16(3):1397–1411, 2016.

[107] Jiangtao Yuan, Jing Yang, Chenyu Wang, Xingxing Jia, Fang-Wei Fu, and Guoai Xu. A new efficient hierarchical multi-secret sharing scheme based on linear homogeneous recurrence relations. *Information Sciences*, 592:36–49, 2022.

[108] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In *International Workshop on Public Key Cryptography*, pages 277–290. Springer, 2004.

[109] Xingzhou Zhang, Yifan Wang, Sidi Lu, Liangkai Liu, Weisong Shi, et al. Openei: An open framework for edge intelligence. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1840–1851. IEEE, 2019.