Blockchain-Based Privacy-Preserving and Accountable Mobile Edge Outsourcing Computing Framework for the Metaverse

Chunpei Li, Chen Liu, Peng Liu, Xianxian Li, Wangjie Qiu, Lei Lei, and Yanli Jin

***Abstract*—Mobile Edge Computing (MEC) enables Metaverse Terminal Devices (MTD) to perform complex tasks, including graphic rendering and physical simulation, by leveraging low-latency outsourced computing. However, existing research has not fully addressed the challenge of establishing an efficient outsourced computing service within an open and dynamic MEC environment that simultaneously ensures privacy and accountability. To address this, we proposes a blockchain-based privacy-preserving and accountable mobile edge outsourcing computing framework for the Metaverse, termed Meta-BMEOC. Specifically, we have designed an outsourcing computing protocol based on smart contracts and threshold secret sharing, enabling MTD to outsource tasks to multiple edge servers while preserving privacy. Furthermore, we have developed an off-chain smart contract protocol based on a Trusted Execution Environment. This protocol is designed to reduce the risk of malicious edge servers colluding to reconstruct the computational tasks of MTD, and it enables accountability for servers that return erroneous results. Additionally, we designed an incentive mechanism to resist malicious attacks and ensure system security and stability. Security analysis and experimental evaluation show that Meta-BMEOC not only ensures the privacy and accountability of outsourced computing but also provides outsourced computing services with lower computational latency.**

***Index Terms*—Metaverse, blockchain, edge outsourced computation, privacy-preserving computation.**

Manuscript received 8 January 2024; revised 8 May 2024; accepted 21 August 2024. Date of publication 29 August 2024; date of current version 21 May 2025. This work was supported in part by the National Natural Science Foundation of China under Grant U21A20474 and Grant 62166004; in part by the Guangxi Science and Technology Major Project under Grant GuikeAA22068070 and Grant GuikeAD21220114; in part by the Guangxi Natural Science Foundation under Grant 2020GXNSFAA297075; in part by the Guangxi Science and Technology Project under Grant 2021AA11006 and Grant GuikeAD21220114; in part by the National Key Research and Development Program of China under Grant 2021YFB2700300; in part by the Beijing Natural Science Foundation under Grant Z230001; and in part by the Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing. The editor coordinating the review of this article was C. Shen. *(Corresponding authors: Xianxian Li; Wangjie Qiu.)*

Chunpei Li, Peng Liu, Xianxian Li, Lei Lei, and Yanli Jin are with the Key Laboratory of Educational Blockchain and Intelligent Technology, Ministry of Education, Guangxi Key Laboratory of Multi-Source Information Mining and Security, Guangxi Normal University, Guilin 541004, China (e-mail: licp@gxnu.edu.cn; liupeng@gxnu.edu.cn; lixx@gxnu.edu.cn; larainelei@hotmail.com; jyl@stu.gxnu.edu.cn).

Chen Liu is with Zhongguancun Laboratory, Beijing 100094, China (e-mail: liuch1027@163.com).

Wangjie Qiu is with the Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, Beihang University, Beijing 100191, China (e-mail: wangjieqiu@buaa.edu.cn).

Digital Object Identifier 10.1109/TGCN.2024.3451513

I. Introduction

THE METAVERSE is not only a virtual digital world offering boundless creative space but also holds the potential to reshape the current form of the Internet. Compared to the traditional Internet, the metaverse provides users with a more immersive and interactive experience [1]. Virtual goods and services can generate new economic value and business models for human society. For instance, users can create characters in the metaverse for social interaction with other users. Merchants can open shops in the metaverse to sell virtual or physical goods, and companies can establish virtual offices for remote work and collaboration. In summary, the metaverse can redefine the current Internet landscape, representing a new trend in the digital world. To support the construction of this virtual digital world, the metaverse is equipped with a plethora of terminal devices, such as virtual reality (VR), augmented reality (AR), and mixed reality (MR) devices. However, these metaverse terminal devices (MTD) are limited by their physical size and, consequently, the computational resources they carry. MTD need help independently handling many intensive computations, such as graphics rendering, physical simulations, and machine learning calculations, among which polynomial computations are a key component.

Typically, MTD can offload complex and computationally intensive tasks to cloud servers because cloud computing boasts vast, scalable computational resources. In academia, the paradigm of offloading computational tasks to cloud servers is called outsourced computation [2], [3], [4]. However, in most instances, cloud servers are distant from MTD. Uploading many computational tasks to these servers may increase latency and network congestion. The rise of edge computing has recently addressed the limitations and gaps of cloud computing [5], [6]. Edge computing is an emerging distributed computing paradigm that extends computational capacity from the cloud to the network edge, nearer to data sources (users). This positioning offers the advantage of providing MTD with low-latency computation and storage services. Mobile Edge Computing (MEC) is a form of edge computing, specifically focused on performing computational processing in mobile network environments [7]. It is worth mentioning that with the development of Open Radio Access Network (O-RAN) technology, mobile network architecture is becoming more flexible and efficient. Users can now easily connect Metaverse applications to edge computing resources of different network providers using open and standardized interfaces and components. This advancement not only simplifies the process but also significantly reduces the difficulty and cost of broadly accessing edge devices (servers) with Metaverse applications.

Although MEC provides significant convenience and efficiency improvements for edge outsourcing computing in the Metaverse, we must also recognize the security challenges and risks that arise from it. Especially in the dynamic and open MEC environment supported by Open Radio Access Networks, issues of privacy protection and security are particularly prominent. Firstly, due to the inherent openness and dynamism of MEC, it is challenging for MTD to form a stable collaborative relationship with edge servers. This leads to concerns among MTD that edge servers may steal privacy-sensitive information contained in computing tasks and results [2], [3], [4]. On the other hand, MTD are also concerned about the reliability and accuracy of computational results. This is because network providers deploying edge devices might, in an attempt to conserve resources, deliberately instruct edge servers to return inaccurate or low-precision computational results [8]. Furthermore, edge servers may also be susceptible to third-party security attacks, leading to the return of erroneous computational results [6]. These security challenges and risks result in MTD not being able to fully trust the edge servers in MEC, and this relationship can even be defined as a ‘zero-trust’ relationship.

In response to these challenges, some researchers are exploring the integration of blockchain technology into the field of edge computing to address the trust issues encountered in MEC outsourcing [3], [6], [8]. Blockchain is a decentralized database technology that maintains data consistency by implementing consensus mechanisms on a global ledger. Known for its immutability, transparency, and high degree of trust [9], this technology offers a potential solution to the lack of trust in edge computing. Nevertheless, certain challenges persist that warrant further investigation. Primarily, the inherent architecture of blockchain results in constrained transaction throughput, potentially leading to scalability limitations in blockchain-based mobile edge computing platforms [10]. Furthermore, despite blockchain’s ability to ensure transactional transparency and immutability, it necessitates enhanced mechanisms for user privacy protection. This becomes particularly crucial in outsourced computing contexts where stringent privacy safeguards are imperative. Although the studies by Li et al. [6], Wang et al. [3], and Lai and Zhao and [8] attempt to address trust and privacy issues in edge computing from different perspectives, they either fail to adequately consider the protection of input and output privacy in computation, or they rely on computationally intensive fully homomorphic encryption techniques. These approaches may not be suitable for resource-constrained MEC environments and could result in significantly high computational latencies. Furthermore, accountability is a key security attribute in client-server architectures [11]. However, achieving accountable MEC outsourcing computation while protecting privacy, especially in scenarios involving collaboration among multiple servers, remains an inadequately discussed and explored area.

In response to the deficiencies in existing research, this paper proposes a blockchain-based privacy-preserving and accountable mobile edge outsourcing computing framework for the Metaverse, termed Meta-BMEOC. Meta-BMEOC consists of a blockchain service layer, an off-chain smart contract layer, and a computing service layer. The blockchain service layer establishes trust between MTD and edge servers. The off-chain smart contract layer, built on trusted execution environments (TEEs), enables the security and privacy of complex business logic and computation, such as device resource scheduling, accountability of malicious servers, and reward distribution. The computing service layer provides MTD with flexible and scalable edge computing resources. Compared to existing research, Meta-BMEOC does not rely on complex cryptographic mechanisms to protect the privacy of outsourced computing tasks. Instead, it cleverly leverages the dynamism and openness of MEC, protecting privacy through collaborative computation by servers from different providers, while offering precise accountability mechanisms. The main contributions of this paper can be summarized as follows:

To protect the privacy of tasks and final results in outsourced computations within a dynamic and open edge environment, we have designed a privacy-preserving outsourcing computation protocol based on smart contracts and threshold secret sharing mechanisms. This protocol allows MTD to outsource complex polynomial computation tasks to multiple edge servers for collaborative computation, utilizing Shamir’s secret sharing technique to preserve privacy in outsourced computations. Additionally, within this protocol, MTD can locally reconstruct and verify the correctness of the final outsourced computation results.

To prevent edge servers from colluding to reconstruct the inputs and final outcomes of outsourced computations, we have developed a secure and privacy-preserving off-chain smart contract protocol based on trusted execution environments (TEEs). This protocol integrates verifiable random functions and threshold signatures, significantly enhancing the scalability of Meta-BMEOC and safeguarding the privacy of outsourced computations. Particularly in instances of erroneous computational results, it empowers users to accurately hold the servers that return incorrect outcomes accountable.

To bolster the security and longevity of Meta-BMEOC, a novel blockchain-driven incentive scheme has been engineered for edge outsourcing computation. This framework actively encourages participants in the Meta-BMEOC ecosystem to adhere to protocol operations with integrity, thus promoting the robust operation of the network via a structured reward and penalty system. Moreover, in the event of malevolent activities, this incentive model is adept at autonomously enacting rigorous punitive actions.

Finally, utilizing an ideal model-based approach, we rigorously analyzed the privacy, accuracy, and accountability of Meta-BMEOC. Extensive simulation experiments were conducted to evaluate its performance. We demonstrated that Meta-BMEOC offers privacy-preserving, low-latency polynomial outsourcing computation services for mobile edge computing. Moreover, it is capable of identifying and holding malicious servers accountable across multiple server environments.

II. Related Work

This section introduces blockchain technology and its supporting research in outsourced computing.

A. Blockchain Technology

Blockchain technology, originating from Satoshi Nakamoto's Bitcoin proposal in 2008 [9], is a decentralized database technology. Its hallmark is a decentralized architecture, independent of centralized authorities for record maintenance and verification [1]. Built on chained data blocks and cryptographic links like hash functions, blockchain ensures data integrity by making recorded data difficult to alter. Today, blockchain extends beyond cryptocurrencies to key roles in IoT [12], edge computing [13], and 5G/6G networks [14].

Smart contracts are automated, programmable protocols deployed on blockchain networks, capable of executing predefined actions automatically when specific conditions are met, thus enabling the widespread application of blockchain technology in various complex business scenarios [15], [16]. Despite their operational efficiency and reduced transaction costs, smart contracts face challenges in privacy protection and efficiency. Wan et al. [17] developed the zk-AuthFeed framework, utilizing an off-chain computation, on-chain verification approach for privacy and authenticity in DApps. Frassetto et al. [10] introduced an off-chain smart contract protocol using trusted execution environments (TEEs) to improve performance and ensure data security.

B. Outsourced Computing

Outsourced computing involves delegating data processing and computational tasks to third-party service providers with more robust computing resources. For instance, Zhang and Wang [2] proposed a verifiable outsourced computing scheme for multi-server collaboration based on threshold secret sharing protocols. Zhao et al. [18] introduced a novel outsourcing algorithm for K-means clustering computation under the integrated space-air-ground IoT environment, utilizing sparse matrix transformations. Similarly, Li et al. [5] designed a novel privacy-preserving location-based service search scheme that enhances the security of outsourced clouds by implementing an encrypted data storage and query model. Additionally, Liu et al. [19] constructed a secure outsourced neural network inference service using lightweight cryptographic primitives. However, these outsourced computing solutions have not yet fully considered how to establish trust relationships between clients and servers in open and dynamically changing mobile edge computing environments. Therefore, their applicability in MEC scenarios remains to be improved.

To address the issue of trust deficits in outsourced computing, numerous studies have proposed innovative solutions by integrating blockchain technology. For example, Guan et al. [20] introduced a polynomial computation outsourcing scheme that combines variants of Horner's method with blockchain. Li et al. [6] designed an accountable and fair outsourcing computation scheme based on blockchain and ElGamal cryptography. Lai and Zhao [8] launched an edge outsourcing computation solution that operates off-chain. Guo et al. [21] provided secure and reliable computing services for user devices through a novel, rewritable blockchain technology. Zhou et al. [15] devised an outsourcing computation method that reduces the computational burden on smart contracts and users, used for the recovery of secret image sharing. Susilo et al. [22] proposed an outsourcing scheme based on redundant computation, utilizing smart contracts and public key encryption. Additionally, Li et al. [13] and Hou et al. [23] each proposed a multi-authority attribute encryption scheme assisted by blockchain and a verifiable attribute encryption system, respectively, both aimed at enhancing the security and privacy of data processing. Furthermore, Wang et al. [24] and Liu and Zhang [25] introduced a smart contract-based outsourced matrix computation scheme and a multifunctional verifiable matrix multiplication computation scheme, respectively. Despite these developments, current research on blockchain-based outsourced computing primarily focuses on the reliability, verifiability, and fairness of the computations, with insufficient attention to the privacy protection of MEC outsourcing tasks and their outcomes.

In the field of privacy-preserving outsourced computing, Wang et al. [3] proposed a privacy-preserving outsourced computing service that combines smart contracts with fully homomorphic encryption. Additionally, Liu et al. [26] developed a privacy-preserving outsourced computing scheme that integrates Intel SGX and blockchain technology. However, these schemes often rely on complex cryptographic techniques or entirely on specific trusted execution hardware to protect data privacy, thereby limiting their applicability in MEC environments.

In summary, constructing an outsourced computing mechanism that can both protect privacy and ensure performance, especially in an open and dynamically changing MEC environment to meet the demands of the metaverse, remains a significant unresolved research challenge.

III. System Model

This section provides an overview of the Meta-BMEOC system model.

A. Meta-BMEOC Architecture

The architecture of Meta-BMEOC, as depicted in Figure 1, consists of two main components: the metaverse terminal devices and the edge server group. The edge server group includes the blockchain service layer, computational service layer, and the off-chain smart contract execution environment. Metaverse terminal devices outsource computing tasks to the computational service layer using off-chain smart contracts, and subsequently verify the accuracy of the final results locally. If errors are detected in the final results, the servers that submitted the erroneous outcomes are held accountable through off-chain smart contracts.

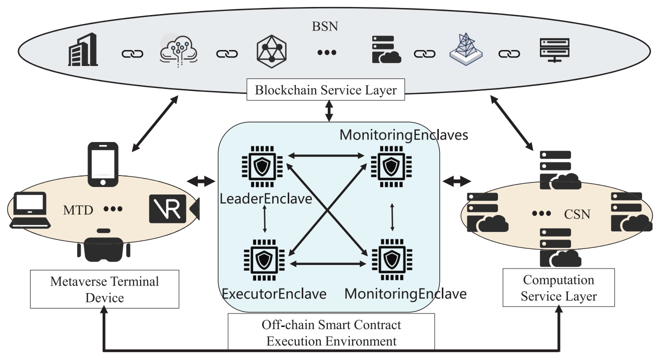

BSN
Blockchain Service Layer
MonitoringEnclaves
LeaderEnclave
ExecutorEnclave
MonitoringEnclave
MTD
CSN
Metaverse Terminal Device
Computation Service Layer
Off-chain Smart Contract Execution Environment

Fig. 1. System Model.

- *Metaverse Terminal Devices (MTD):* MTDs serve as hardware interfaces for accessing the metaverse, using VR, AR, and MR technologies. These devices, limited in computational and storage capacities, connect to the Meta-BMEOC network via options like 4G/5G and WiFi. MTDs with low computational power require gateway devices for network connection.
- *Edge Server Group (ESG):* In the Meta-BMEOC architecture, ESG comprises edge servers providing stable computational resources to MTDs. Some servers support trusted execution environments (TEEs) like Intel SGX or ARM TrustZone, and are deployed across public and private clouds, and base stations, offering low-latency services. ESG is divided into three layers: Blockchain Service Layer, Computational Service Layer, and Off-chain Smart Contract Execution Environment, each with distinct responsibilities.
- *Blockchain Service Layer (BSL):* BSL, forming the trust foundation within Meta-BMEOC, consists mainly of Blockchain Consensus Nodes (BCN). BCNs execute consensus protocols ensuring data consistency and system security. Servers in TEEs and the computational layer act as blockchain clients interacting with BSL, maintaining system security and stability through decentralized incentives.
- *Computation Service Layer (CSL):* Composed of Computational Service Nodes (CSNs), CSL offers scalable computing resources to MTDs. CSNs collaborate to execute computation-intensive tasks while ensuring security and privacy. Inputs $\boldsymbol{x}$ and functions F are divided and processed in parallel, optimizing CSNs' computational resources. MTDs reconstruct complete results from the partial outputs of each CSN.
- *Off-chain Smart Contract Trusted Execution Environment (OSCTEE):* OSCTEE is a TEE-based smart contract execution platform that integrates verifiable random functions (VRF) and threshold signatures technology, designed to execute off-chain smart contracts securely and efficiently. Leveraging the isolation capabilities of TEEs, OSCTEE not only enhances execution efficiency but also strengthens data privacy protection. In the event of errors in outsourced computations, MTDs can utilize OSCTEE to recalculate, accurately identifying and holding the responsible CSNs accountable.

Please note that Meta-BMEOC, by dividing edge servers into different groups, necessitates special attention to the allocation and optimization of distributed computing resources during practical deployment. Essentially, Meta-BMEOC's resource allocation strategy deals with the optimization of multi-user and multi-server interactions under the premises of ensuring security and privacy protection. However, due to space constraints, the detailed optimization strategies for Meta-BMEOC resource allocation will be discussed in another paper. Additionally, readers interested in this topic can refer to another paper [27] by our team, which explores resource allocation optimization for multi-user, multi-server computation offloading in edge computing.

To ensure the seamless integration of Meta-BMEOC within the metaverse ecosystem, we recommend adopting the inter-blockchain communication (IBC) protocol from Cosmos and the Substrate technology from Polkadot, facilitating an effective connection between Meta-BMEOC and the existing blockchain systems in the metaverse. This approach facilitates data format conversion, protocol adaptation, and API calls, supporting developers in creating applications that can be ported across different blockchain platforms. If Meta-BMEOC is deployed as an independent blockchain system, we recommend using cross-chain protocols such as IBC, ChainBridge, and ChainLink to facilitate asset movement and information exchange with other blockchain systems in the metaverse.

B. Attack Model

Meta-BMEOC aims to protect the security and privacy of MTD's outsourced tasks against two primary types of edge servers:

- *Semi-Honest Edge Servers:* These servers adhere to Meta-BMEOC's protocol but may try to analyze accessible data to glean confidential information. They do not alter or spread false data.
- *Malicious Edge Servers:* These servers can blatantly disregard Meta-BMEOC's protocols and may engage in activities that disrupt computation processes, alter results, or compromise MTD data privacy.

C. Security Assumptions

Both semi-honest and malicious edge servers are assumed to have limited computational resources, preventing them from breaking Meta-BMEOC's encryption within polynomial time. They are incapable of generating fake digital signatures, reversing hash values to original messages, or decrypting ciphertexts to reveal plaintexts. It is assumed that Meta-BMEOC's message transmission channels are secure and reliable, using protocols like TLS/SSL for end-to-end data encryption and entity authentication.

To ensure that the Meta-BMEOC system can adapt to various practical application needs, we assume that the entities participating in this system will choose appropriate key storage security measures based on their specific conditions [28], [29].

TABLE I
KEY MATHEMATICAL SYMBOLS USED IN THE META-BMEOC

Symbol	Meaning	Symbol	Meaning
λ	ECC security parameter	sk	Private key
pk	Public key	$j, i, \jmath, \imath$	Indices, used to denote specific entities
M	Total number of CSNs	TX	Blockchain transaction
ID	Unique identifier	H	Hash function
$\boldsymbol{x}$	Vector representation of the computational task	F	Polynomial function
k	Number of variables in polynomial F	m	Number of CSNs selected for the task
t	Secret sharing security threshold	d	Degree of polynomial F
α	Verification key for computation results	r_i, γ_i	Random vectors and scalars
$v(\mu), w(\mu)$	Polynomials constructed to distribute shares of $\boldsymbol{x}$ and α	$\partial_\jmath$	Input share for $CSN_\jmath$
$T_\jmath$	Task received by $CSN_\jmath$	$f_\jmath$	Polynomial function assigned to $CSN_\jmath$
$c_\jmath$	Preliminary result calculated by $CSN_\jmath$	$b_\jmath$	Product of $c_\jmath$ and $w(\jmath)$
$R_\jmath$	Computation result generated by $CSN_\jmath$	$\phi(\mu), \psi(\mu)$	Polynomials for reconstructing the result

These measures include, but are not limited to, hardware security modules (HSM), trusted platform modules (TPM), and distributed storage mechanisms. The design of these security measures aims to address different security threat scenarios and effectively mitigate the risks associated with key loss or leakage. Through this flexible selection of security strategies, Meta-BMEOC aims to enhance the overall resilience and security of the system.

Additionally, although Meta-BMEOC employs current cryptographic algorithms to ensure system security, it is imperative to recognize that with advances in computing technology and the development of quantum computing, existing encryption technologies may face the risk of being compromised. Therefore, we recommend considering the adoption of more advanced encryption technologies, such as lattice-based cryptographic algorithms, to enhance the system's security protection when using Meta-BMEOC.

IV. CORE COMPONENTS OF META-BMEOC

In this section, we will provide a detailed introduction to Meta-BMEPC's outsourced computing protocol, off-chain smart contract protocol, and incentive mechanism. The key mathematical symbols involved in Meta-BMEOC and their meanings are shown in Table I.

A. Privacy-Preserving Outsourced Computing Protocol

A fundamental objective of Meta-BMEOC is to provide trustworthy and privacy-protective edge outsourced computing services to MTD in a zero-trust environment. However, in the open and dynamic edge computing context, no participant, including CSNs, can be fully trusted. To address this, we have designed a privacy-preserving outsourced computing protocol based on blockchain technology and threshold secret sharing protocols. This approach leverages the blockchain to establish trust among participants and uses threshold secret sharing to safeguard the privacy of computational tasks and their final outcomes. Additionally, MTD can locally verify the accuracy of the final computational results. In cases of discrepancies, they can hold malicious edge servers accountable through smart contracts (operating based on the off-chain smart contract protocol discussed in the following section).

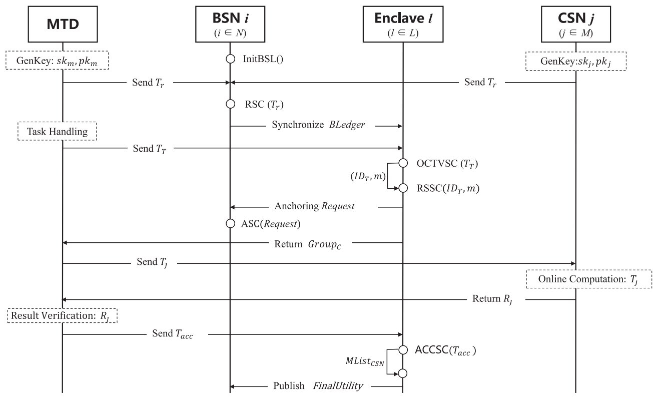


Fig. 2. The sequence diagram of TEMPC.

Compared to Zhang and Wang approach [2], Meta-BMEOC's outsourced computing protocol integrates blockchain and smart contract technologies, making it not only suitable for the open and dynamic MEC environment but also capable of precisely holding malicious servers accountable. Notably, in Meta-BMEOC, the verification of the final computation results' correctness and the accountability of malicious servers are decoupled. This decoupling implies that accountability is a post-event protocol. As a result, under normal conditions, this design does not increase the resource overhead for outsourced computing (online computation), effectively ensuring the performance and efficiency of Meta-BMEOC's outsourced computing.

The formalized process of the Meta-BMEOC outsourced computing protocol encompasses Initialization, Registration, Task Handling, Online Computation, Result Verification and Reconstruction, and Accountability. Next, we detail the design of this protocol using an example involving one MTD, N BCNs, and M CSNs. Figure 2 illustrates the sequence diagram of this protocol.

1) Initialization: This step primarily entails the generation of blockchain key pairs and initiation of the blockchain network.

i) Generation of Key Pairs: Using elliptic curve cryptography, participants (MTDs, BCNs, CSNs) generate key pairs (sk, pk) with the algorithm $KeyGen(\lambda)$, where λ is a security parameter. The private key sk_i, confidential to each participant, is used for signature creation and decryption. The public key pk_i, known publicly on the blockchain ledger, is used for signature verification and encryption. It is advisable to create two separate key pairs: one for encryption/decryption and another for digital signatures, enhancing security by reducing single key vulnerability risks.

ii) Launching the Blockchain Network: Considering key factors such as transaction throughput, privacy protection, and cost, Meta-BMEOC opts to employ a permissioned consortium blockchain (such as Hyperledger Fabric, Quorum, ChainMaker) to construct its blockchain network. During the initialization phase, Meta-BMEOC starts by selecting a limited number of trusted edge nodes to form a consensus committee. The selection of these nodes may be based on factors such as the reputation of the node owners (e.g., major telecommunications operators) or their collateral assets. The Byzantine fault-tolerant consensus protocol recommended by Meta-BMEOC aims to process transactions, ensuring the system's fault tolerance and consistency, and achieving high transaction processing efficiency. Assume N is the total number of candidate nodes for PBFT, and $n \subset N$ represents the selected set of members for the consensus committee. The reputation and collateral assets of each node $i \in n$ are denoted by η_i and ξ_i, respectively. The consensus process can then be expressed as follows:

$$C(n) = \bigoplus_{i \in n} (\theta \eta_i + \omega \xi_i). \tag{1}$$

Here, $\bigoplus$ represents the collective operation for achieving consensus within the committee through PBFT, while θ and ω are the weighting coefficients for reputation and collateral assets, respectively. Furthermore, given the openness and dynamism of the edge environment, traditional Byzantine fault-tolerant consensus mechanisms may not fully meet the needs of the Meta-BMEOC system. Therefore, Meta-BMEOC plans to integrate with the latest research outcomes to realize a system that is secure, decentralized, and features low latency. These studies employ deep reinforcement learning paradigms to help the consensus mechanism adapt to the dynamic changes in the IoT edge environment, particularly in how to effectively form and maintain a consensus committee in dynamic settings, as well as ensuring the efficiency and robustness of blockchain transactions. Let $\mathcal{D}$ represent the dynamic state of the environment, and Υ denote the consensus strategy adopted under a given state. The state transition function is represented by $\Phi(\mathcal{D}_{\tau+1} | \mathcal{D}_\tau, \Upsilon_\tau)$, and the reward function by $\Lambda(\mathcal{D}_\tau, \Upsilon_\tau)$ (for example, maximizing the throughput of blockchain transactions while ensuring security and decentralization). The dynamic optimization strategy for Meta-BMEOC can then be expressed as follows:

$$\Upsilon^*(\mathcal{D}) = \arg\max_{\Upsilon} \mathbb{E}\left[\sum_{\tau=0}^{\infty} \sigma^\tau \Lambda(\mathcal{D}_\tau, \Upsilon(\mathcal{D}_\tau))\right]. \tag{2}$$

Here, σ is the new discount factor, and $\mathbb{E}$ represents the expected operation under the strategy Υ. Due to space limitations, for more detailed discussions and related works, please refer to our previous research publications [30] and other relevant studies. After the formation of the consensus committee, the blockchain consensus network is initiated through the creation of the genesis block. The genesis block contains essential details such as the list of consensus committee members, the list of legitimate enclaves, and the initial token distribution scheme. It is worth noting that token creation and management can be implemented through smart contracts or by referencing standards such as ERC-20 and BEP-20.

2) Registration: When initializing the blockchain network, both MTD and CSN_j (where $j \in [M]$) must register. MTD registers with a transaction $TX_r = (ID, ID_M, pk_M, Sig_{T_r}, Other)$, where TX_r is the transaction type, *ID* is its identifier, ID_M is MTD's unique blockchain identity (derived as $ID_M = H(pk_M)$ using a secure hash function $H()$, pk_M is MTD's ECC public key, Sig_{TX_r} is its ECC digital signature, and *Other* includes additional attributes like timestamps. The BCN verifies MTD's registration via a smart contract.

CSN_j's registration is similar, with its *Other* field containing a tuple $(IP, R_{CPU}, R_{RAM}, R_{Storage}, R_{Network})$ indicating its Internet address and computational, storage, and network resources. Post-registration, all entities must acquire a certain amount of tokens for future rewards or indemnifications related to outsourced computations.

3) Task Handling: After completing the initialization and registration processes, the system officially enters the outsourcing computation phase of Meta-BMEOC. In this phase, the MTD employs Shamir's Secret Sharing protocol to divide complex computation tasks into multiple subtasks, which are then outsourced to different CSNs) through the blockchain for processing. Once these nodes complete the computations, they return partial results, and the MTD subsequently carries out local verification and reconstruction of the results. If errors are detected in the computation outcomes, the MTD utilizes off-chain smart contracts to identify and hold accountable those CSNs that returned incorrect results.

Specifically, Let $\boldsymbol{x} \in \mathbb{F}_q^k$ be the computational input and F be a d-degree polynomial representing the outsourced computational function, where k signifies the number of variables in the polynomial. For any $\jmath \in [m] \leq [M]$, the MTD directly sets $f_\jmath = F$ as its function share.

For the computation input $\boldsymbol{x}$, the MTD stochastically selects $\alpha \in \mathbb{F}_q^*$, vectors $\mathrm{r}_1, \ldots, \mathbf{r}_t \in \mathbb{F}_q^k$, and scalars $\gamma_1, \ldots, \gamma_t \in \mathbb{F}_q$. Subsequently, by employing Shamir's secret sharing scheme [26], the MTD distributes shares of $\boldsymbol{x}$ and α to $CSN_\jmath$. The MTD constructs a t-degree polynomial as:

$$v(\mu) = \boldsymbol{x} + \sum_{i=1}^{t} \mathrm{r}_i \mu^i, \tag{3}$$

$$w(\mu) = \alpha + \sum_{i=1}^{t} \gamma_i \mu^i. \tag{4}$$

For any $\jmath \in [m], m \leq M$, MTD calculates:

$$\partial_\jmath = (v(\jmath), w(\jmath)). \tag{5}$$

where $\partial_\jmath$ denotes the input share allocated for $CSN_\jmath$. Herein, α serves as the validation key and is securely retained by the MTD.

Upon completing the above procedures, the MTD consecutively performs hash computations to obtain $ID_{T_j} = H(f_j, \partial_j)$, resulting in the task identifier $ID_t = (ID_{T_1}, ID_{T_2}, \ldots, ID_{T_m})$, for $j \in [m]$. It then crafts a transaction as delineated below, sending it to the off-chain smart contract execution environment:

$$TX_t = \big(ID, ID_T, ID_M, Dep., m, Sig_{T_T}, Other\big). \tag{6}$$

where TX_t represents an outsourced computation transaction and *Dep.* is the deposit for this computational task, designated to compensate the honest CSN. Upon receiving the transaction, the LeaderEnclave verifies the legitimacy of the outsourced computation through the outsourced computation transaction verification smart contract. Once approved, the leaderEnclave selects *m* servers (with $m \leq M$) from the CSNs list to form the outsourcing CSNs group, $Group_C$, and communicates it to the MTD. This is done through the resource scheduling smart contract. Additionally, the digital assets of the relevant participants are anchored through the anchoring smart contract. Finally, the MTD dispatches the outsourced computational task, denoted as $T_j = (ID_{T_j}, f_j, \partial_j)$, to the corresponding CSN_j.

4) Online Computation: Once CSN_j receives the task T_j, it first verifies the validity of T_j using digital signature algorithms. If the verification is successful, CSN_j uses ∂_j and f_j as inputs and carries out the subsequent computations:

$$c_j = f_j(v(j)), \tag{7}$$
$$b_j = c_j \times w(j). \tag{8}$$

After completing the computations, CSN_j generates a signature $Sig_{T_j} = Sign(sk_{C_j}, c_j, b_j)$. Then, it forms the computational result $R_j = (ID_{T_j}, c_j, b_j, Sig_{T_j})$ and sends it back to MTD.

5) Result Verification & Reconstruction: Upon receiving all $R_j = (ID_{T_j}, c_j, b_j, Sig_{T_j})$ from the MTD, the first step is to execute $Ver(pk_{c_j}, c_j, b_j, Sig_{T_j})$ to verify the authenticity of the signature. Once verified, the MTD reconstructs two polynomials, $\phi(\mu)$ and $\psi(\mu)$, using interpolation techniques. Furthermore, the degree of $\phi(\mu)$ is less than or equal to *dt*, and the degree of $\psi(\mu)$ is less than or equal to $(d + 1)t$. For every j, it holds that $\phi(j) = c_j$ and $\psi(j) = b_j$.

MTD then determines, using a predefined verification key α, whether the following equation is satisfied:

$$\psi(0) = \alpha\phi(0). \tag{9}$$

If the Equation (9) is satisfied, the MTD outputs $\phi(0)$, indicating successful verification; otherwise, it outputs $\perp$, signifying a failed verification. The fundamental principle of this process is that without the knowledge of α, it is challenging for a malicious CSN to satisfy Equation (9). Upon successful verification, the MTD uses interpolation based on c_j to formulate the polynomial $\phi(\mu) = F(v(\mu))$, which represents the final result obtained by the MTD. If no further actions are taken, the OSC contract automatically allocates its deposit to the CSNs engaged in the computational task.

Algorithm 1 ACC Algorithm

```
Require: Accountability transaction $TX_{acc}$ = $(ID, ID_T, R_j, T_j, Sig, Other)$ where $j \in [m] = [1, \cdots, m]$.
Ensure: If malicious CSN is detected, return the list $MList_{CSN}$. Otherwise, return 0.
 1: function ACC($T_{acc}$)
 2:     if not isTransaction($T_{acc}$) OR not isPublicKeyConsistent($T_{acc}$) then
 3:         return 0
 4:     end if
 5:     $MList_{CSN} \leftarrow$ detectMaliciousCSN($R_j$, $T_j$)
 6:     if $MList_{CSN}$ is not empty then
 7:         return $MList_{CSN}$
 8:     else
 9:         return 0
10:     end if
11: end function
12: function DETECTMALICIOUSCSN($R_j$, $T_j$)
13:     Initialize $MList_{CSN} = []$.
14:     for $j = 1$ to $m$ do
15:         Extract $R_j = (ID_T, c_j, b_j)$ and $T_j = (ID_T, f_j, \partial_j, pk_M, Sig_T)$ where $\partial_j = (v(j), w(j))$
16:         $c'_j \leftarrow F(v(j))$
17:         $b'_j \leftarrow c_j w(j)$
18:         if $c'_j \neq c_j$ or $b'_j \neq b_j$ then
19:             Identify $ID_{C_j}$ using public key corresponding to $(c_j, b_j)$
20:             Append $ID_{C_j}$ to $MList_{CSN}$
21:         end if
22:     end for
23:     return $MList_{CSN}$
24: end function
```

6) Accountability: If the final computation results fail to pass the validation, MTD will identify the malicious CSN through the smart contract *ACC* and penalize them. Specifically, MTD constructs a public accountability transaction and sends it to the *ACC*:

$$TX_{acc} = \big(ID, ID_T, R_j, T_j, Sig, Other\big). \tag{10}$$

where $j \in [m] = [1, \ldots, m]$.

Upon receiving TX_{acc}, *ACC* first validate the transaction's integrity. It then performs the following operation to obtain a list of malicious CSNs:

$$MList_{CSN} = ACC(TX_{acc}). \tag{11}$$

ACC represents the accountability smart contract, as illustrated in **Algorithm 1**. After acquiring the malicious CSN list $MList_{CSN}$, *ACC* will automatically deduct related tokens from the accounts of the malicious CSN to reward other CSNs.

B. Off-Chain Smart Contracts Protocol

In the previous section, we thoroughly introduced the Meta-BMEOC privacy-preserving outsourcing computation protocol. This protocol relies on blockchain technology and threshold secret sharing to ensure the trustworthiness and privacy of outsourced computations in an open and dynamic edge environment. However, this design faces two key challenges: First, due to the limitations of consensus protocols and resources, traditional blockchain smart contracts are insufficient in performance, struggling to meet the high-performance demands of outsourced computations and certain complex calculations. Second, using the threshold secret sharing protocol for privacy protection might inadvertently facilitate collusion among malicious servers via the blockchain's public ledger, leading to the reconstruction of secret information. To address these challenges, this section introduces a TEE-based off-chain smart contract protocol. Compared to smart contract protocols that solely rely on TEE technology [10], this protocol integrates verifiable random functions and threshold signatures technologies, aiming to achieve a fairer, safer, and more decentralized off-chain smart contract execution framework in an open and dynamic edge environment.

The off-chain smart contract protocol primarily defines an execution environment comprised of multiple TEE enclaves, which facilitates the operation of off-chain smart contracts. This environment includes a Leader Enclave, an Execution Enclave, and several Monitoring Enclaves. The specific execution process is as follows:

1) Registration: Enclaves participating in the off-chain smart contract protocol must obtain a legal identity through a registration process. Specifically, first, the enclave E_i selects n_s coprime positive integers $s_1, s_2, \ldots, s_{n_s}$ as seeds. Then, E_i uses the generator g_1 of the multiplicative cyclic group $\mathbb{G}_1$ to generate the key triplet $(pk_i^{agg}, sk_i^{agg}, x_i)$. Here, the private key sk_i^{agg} is a secure random number, and the public key pk_i^{agg} is computed as $g_1^{sk_i^{agg}} \mod p$, where p is a prime number. The private value x_i, used for bilinear pair signatures, and its corresponding public value $g_1^{x_i}$ will be used in the signature aggregation and verification process. Threshold signatures require a mapping e, which maps the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ to $\mathbb{G}_3 : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$. Additionally, h_h is a homomorphic hash function that satisfies $h_h(\mathrm{cnf}_1 + \mathrm{cnf}_2) = h_h(\mathrm{cnf}_1) \oplus h_h(\mathrm{cnf}_2)$, where $\oplus$ represents an operation ensured by the homomorphic property of h_h.Next, E_i generates a proof $\pi_{sk_i^{agg}, x_i}$ through the TEE protocol, verifying its private control over (sk_i^{agg}, x_i), and submits $(\pi_{sk_i^{agg}, x_i}, pk_i^{agg}, g_1^{x_i})$ to the on-chain management smart contract (MSC) to complete the registration. Once registered, the MSC sends the latest block data to E_i. E_i verifies the received block information and returns a proof π_b containing the block hash and height. Upon completion of these steps, the MSC adds E_i to the list of legitimate enclaves.

2) Execution Environment Creation: Users send requests to the MSC, utilizing a erifiable random function (VRF) to select n_e registered enclaves E from a list of legitimate enclaves, thus forming the off-chain smart contract execution environment $\mathcal{E}$. Initially, each enclave E_i generates a VRF key pair (sk_{v_i}, pk_{v_i}). Then, using a common seed $\mathscr{S}$, E_i generates VRF proof materials: $\beta_i \leftarrow \mathrm{VRF.hash}(sk_{v_i}, \mathscr{S})$, and $d_i, \pi_i^v \leftarrow \mathrm{VRF.prove}(sk_{v_i}, \mathscr{S})$. To facilitate the identification of the execution environment $\mathcal{E}$, the seed $\mathscr{S}$ should correspond to the identifier of the environment $\mathcal{E}$. E_i sends (β_i, d_i, π_i^v) to the MSC. The MSC verifies the validity of $\mathrm{VRF.verify}(\beta_i, d_i, \pi_i^v, \mathscr{S})$. Upon successful verification, the MSC sorts all d_i in ascending order: $d_{(1)}, d_{(2)}, \ldots, d_{(n)} = \mathrm{sort}(d_1, d_2, \ldots, d_{n_s})$. The node corresponding to the smallest d_i acts as the Leader Enclave, the second smallest as the Execution Enclave, and the remaining nodes as Monitoring Enclaves. The Leader Enclave is responsible for managing the execution pool, deploying smart contracts, and communicating with the on-chain environment. The Execution Enclave executes contract operations and reports state updates to the Monitoring Enclaves, which are responsible for verifying and monitoring these updates. In this manner, an off-chain smart contract execution environment comprising k enclaves is successfully established.

3) Contract Installation: After the execution environment is established, the user first submits the smart contract program *SC* to the Management Smart Contract (MSC) for registration. The MSC assigns a unique identifier id_{sc} to this new contract. Subsequently, the user sends a creation request containing the *SC* program code to the Leader Enclave. Upon receiving the request, the Leader Enclave is responsible for distributing *SC* to the other enclaves within the execution environment $\mathcal{E}$. In $\mathcal{E}$, each enclave locally installs *SC* and generates a signature based on the installation confirmation message cnf_i, where $Sig_i^{\mathrm{d}} = (h_h(\mathrm{cnf}_i))^{x_i}$, and then sends it to the Leader Enclave. During this process, the Leader Enclave performs the following operation to aggregate the received signatures:

$$e\Big(g_1, \mathrm{Sig}^{\mathrm{d}}\Big) = e\left(g_1, \prod_{i=1}^{k} \mathrm{Sig}_i^{\mathrm{d}}\right). \tag{12}$$

The Leader Enclave sends the aggregated signature and confirmation message to the MSC. The MSC first uses a Verifiable Random Function to confirm that the enclaves that have installed the smart contract *SC* belong to a legitimate execution environment, specifically $\mathrm{VRF.verify}(\beta_i, d_i, \pi_i^v, \mathscr{S})$. Upon successful verification, the MSC validates whether all enclaves in the execution environment have correctly installed the contract *SC* using the following equation:

$$\begin{aligned} e\Big(g_1, \mathrm{Sig}^{\mathrm{d}}\Big) &= e\left(g_1, \prod_{i=1}^{k} h_h(\mathrm{cnf}_i)^{x_i}\right) \\ &= \prod_{i=1}^{k} e\big(g_1, h_h(\mathrm{cnf}_i)^{x_i}\big) \\ &= \prod_{i=1}^{k} e\big(g_1^{x_i}, h_h(\mathrm{cnf}_i)\big) \\ &= e\left(\prod_{i=1}^{k} g_1^{x_i}, \prod_{i=1}^{k} h_h(\mathrm{cnf}_i)\right). \end{aligned} \tag{13}$$

If the validation fails, the MSC initiates a challenge process, the details of which will be observed during the challenge-response procedure.

4) Contract Execution Process: After the contract installation is complete, users can send execution requests to the

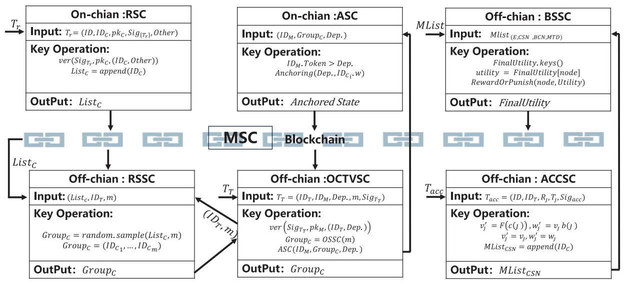
Execution Enclave. The Execution Enclave performs the tasks according to the input and specified operations of the smart contract SC, and generates an execution signature $\mathrm{sig}_{\mathrm{e}}^{\mathrm{o}}$ using aggregate signature technology. Then, it securely propagates the updated state $S_{\mathrm{e}}^{\mathrm{o}}$ to the Monitoring Enclave and the Leader Enclave. In the process of executing the same request, the Monitoring Enclave also generates a corresponding aggregate execution signature $\mathrm{sig}_{\mathrm{s}}^{\mathrm{o}}$ and updates the state $S_{\mathrm{s}}^{\mathrm{o}}$, subsequently sending this information to the Leader Enclave. The Leader Enclave first checks whether the post-execution states provided by the Execution and Monitoring Enclaves are consistent, i.e., it verifies $S_{\mathrm{e}}^{\mathrm{o}} \overset{?}{=} S_{\mathrm{s}}^{\mathrm{o}}$. Once the states are confirmed to be consistent, the Leader Enclave further verifies the validity of the threshold signatures. If all checks and verifications pass, the Leader Enclave confirms that the execution request has been correctly performed and synchronizes the latest state with the MSC when necessary.

5) Challenge and Response: During the execution process of off-chain smart contracts, if any party (either a user or an enclave) fails to timely receive a response from the other side, a challenge will be initiated on the blockchain against the non-responsive party. Specifically, if the Execution Enclave or the Monitoring Enclave fails to respond within a predetermined time frame, a challenge-response mechanism is triggered to compel the relevant enclave to provide the necessary response. The initiating party launches the challenge process through the blockchain. Once the MSC receives a challenge, it is responsible for verifying the validity of the challenge message and recording it accordingly. If the challenged party (whether it is the Execution Enclave or the Monitoring Enclave) fails to provide a satisfactory response within the stipulated time frame, they will be removed from the execution pool and subjected to appropriate penalties. This measure is designed to ensure the smooth operation and integrity of the system.

Although TEEs technology provides a secure execution environment for relevant computational tasks, we must recognize that the security of TEEs is not flawless. Due to possible discrepancies between the security assumptions of TEEs and actual application scenarios, they may be susceptible to threats from software, architecture, and memory attacks. Therefore, it is advisable to implement multi-layer security measures when using TEEs technology, including malware detection, formal verification, environmental isolation, and physical memory control, to enhance the overall security of the system [31]. Additionally, it should be noted that not all servers support the TEEs environment. TEEs also face limitations such as the single processor monopoly, which restricts the full utilization of host computational resources [32]. Consequently, it is recommended that the use of TEEs be limited to critical tasks [33], such as managing privacy resource scheduling in the Meta-BMEOC system and accountability for malicious servers.

C. Meta-BMEOC Smart Contracts

In the Meta-BMEOC system, off-chain smart contracts primarily consist of outsourced transaction verification contracts, resource scheduling contracts, and accountability contracts. To elaborate in detail on the operational mechanisms of these contracts, Figure 3 enumerates the major smart contracts involved in Meta-BMEOC, covering both on-chain and off-chain components.


On-chian :RSC
T_r
Input: $T_r = (ID, ID_C, pk_C, Sig_{\{T_r\}}, Other)$
Key Operation:
$ver(Sig_{T_r}, pk_C, (ID_C, Other))$
$List_C = append(ID_C)$
OutPut: $List_C$
On-chian :ASC
Input: $(ID_M, Group_C, Dep.)$
Key Operation:
$ID_M.Token > Dep.$
$Anchoring(Dep., ID_{C_i}.w)$
OutPut: Anchored State
$MList$
Off-chian : BSSC
Input: $Mlist_{(E,CSN,BCN,MTD)}$
Key Operation:
$FinalUtility.keys()$
$utility = FinalUtility[node]$
$RewardOrPunish(node, Utility)$
OutPut: $FinalUtility$
MSC
Blockchain
$List_C$
Off-chian : RSSC
Input: $(List_c, ID_T, m)$
Key Operation:
$Group_C = random.sample(List_C, m)$
$Group_C = (ID_{C_1}, \dots, ID_{C_m})$
OutPut: $Group_C$
(ID_T, m)
T_T
Off-chian :OCTVSC
Input: $T_T = (ID_T, ID_M, Dep., m, Sig_{T_T})$
Key Operation:
$ver\left(Sig_{T_T}, pk_M, (ID_T, Dep.)\right)$
$Group_C = OSSC(m)$
$ASC(ID_M, Group_C, Dep.)$
OutPut: $Group_C$
T_{acc}
Off-chian : ACCSC
Input: $T_{acc} = (ID, ID_T, R_J, T_J, Sig_{acc})$
Key Operation:
$v_J' = F(c(J)), w_J' = v_J b(J)$
$v_J' = v_J, w_J' = w_J$
$MList_{CSN} = append(ID_C)$
OutPut: $MList_{CSN}$
Fig. 3. Smart Contracts in Meta-BMEOC.

- *On-Chain Smart Contracts:* The on-chain component of the Meta-BMEOC system primarily encompasses two core types of smart contracts: registration smart contracts (RSC) and anchoring smart contracts (ASC). The RSC is responsible for providing identity registration functions for both the MTD and CSNs. In contrast, anchoring smart contracts serve as the principal communication bridge between the on-chain and off-chain systems, specifically designed to anchor and synchronize the status of on-chain assets.
- *Off-Chain Smart Contracts:* The off-chain component of the Meta-BMEOC system mainly comprises four core categories of smart contracts: outsourced computation transaction verification smart contracts (OCTVSC), resource scheduling smart contracts (RSSC), accountability smart contracts (ACCSC), and settlement smart contracts (SSC). First, the OCTVSC is responsible for verifying the validity of outsourced computational transactions submitted by MTD. Upon successful verification, it triggers the RSSC to allocate an adequate number of idle CSNs for executing the outsourced computation tasks. Second, the ACCSC identifies and penalizes malicious CSNs based on the accountability transactions submitted by MTD, subsequently forwarding the list of malicious CSNs to the Settlement Smart Contract. Finally, the SSC implements anti-malicious attack incentive mechanisms based on the received list of malicious nodes, which may include CSNs, BCNs, and enclaves, thereby administering corresponding token rewards or penalties.

D. Anti-Malicious Attack Incentive Mechanism

We introduce an anti-malicious attack incentive mechanism in the Meta-BMEOC architecture to enhance the security and sustainability of outsourced computation. This mechanism is based on blockchain technology and smart contracts, ensuring transparency and tamper-resistance. A key assumption of this mechanism is the use of the Byzantine Fault Tolerance (BFT) consensus algorithm in the blockchain service layer, as discussed in [34], which effectively identifies and isolates Byzantine faulty nodes. Our incentive mechanism relies on the accurate detection and penalization of malicious nodes.

Let X_l represent the total number of nodes of a certain type (such as BCN, CSN, or Enclave), and $\mathscr{X}_{l_i}$ denote the basic staked token amount for node l_i. Additionally, $\mathscr{B}_{mtd}$ refers to the reward paid by MTD for one outsourced computation to the corresponding node. R_l is the additional reward coefficient, and T_l records the number of consecutive honest behaviors, serving as the primary indicator of a participant's reputation. P_l represents the probability of successfully detecting malicious behavior, D_l is the accuracy of this detection, and $\mathcal{X}_l$ indicates the number of anomalous nodes within that node type. ϱ is the penalty multiplier for malicious behavior.

The utility function $U_{l_i}(a_{l_i})$ represents the utility gained by node l_i based on its behavior a_l, and is expressed as follows:

$$U_{l_i}(a_l) = \begin{cases} \mathscr{B}_{mtd} + R_{l_i} \cdot T_{l_i} + \frac{P_l \cdot D_l \cdot X_l \cdot \mathscr{X}_{l_i}}{X_l - \mathcal{X}_l} & \text{if } a_l = a_l^1, \\ -\varrho \cdot \left(\mathscr{B}_{mtd} + \mathscr{X}_{l_i}\right) & \text{if } a_l = a_l^2. \end{cases} \tag{14}$$

Here, a_l^1 denotes the behavior where node l faithfully executes its duties, while a_l^2 represents malicious or Byzantine failure behaviors exhibited by the node. Moreover, different values of R_l and T_l can be set for different types of nodes, such as BCNs, CSNs, and Enclaves, to enhance the flexibility of the incentive mechanism.

To help readers understand the incentive mechanism, we have designed a token for the Meta-BMEOC system named "MetaCoin" (token symbol: MTC), with a total supply set as a constant $T^{MTC} = 1,000,000,000$ MTC. These tokens are allocated for network operation, development, rewarding early contributors, and incentivizing participation in outsourced computing tasks:

$$T^{MTC} = T_D^{MTC} + T_I^{MTC} + T_C^{MTC} + T_P^{MTC}. \tag{15}$$

where T_D^{MTC}, T_I^{MTC}, T_C^{MTC}, T_P^{MTC} represent the amounts of MTC allocated to network operation and development, early contributors and investors, community construction, and outsourcing computation incentives, respectively. M_D^{MTD} is used to reward participants who consistently demonstrate honest behavior, serving as mining rewards to maintain the security and stability of the Meta-BMEOC network. Additionally, Meta-BMEOC utilizes cross-chain technology to enable interconnectivity with existing cryptocurrency systems, enhancing the system's security and broadening its application scope.

V. System Analysis

In this section, we delve into the correctness, privacy, security, and accountability of Meta-BMEOC.

A. Correctness

In Meta-BMEOC, let $\boldsymbol{x} \in \mathbb{F}_q^m$ be the input and F the computational function. MTD randomly selects $\alpha \leftarrow \mathbb{F}_q^*$ and constructs polynomials $v(\mu)$ and $w(\mu)$ to distribute $\boldsymbol{x}$ and α to CSNs. When each CSN computes $c_i = F(v(i))$ and $b_i = c_i w(i)$, MTD uses interpolation to derive $\phi(\mu)$ and $\psi(\mu)$ satisfying $\phi(i) = c_i$ and $\psi(i) = b_i$. In the result verification phase, MTD employs polynomial interpolation to derive $\phi(\mu)$ and $\psi(\mu)$. According to their construction:

1) $\phi(0) = F(v(0)) = F(\boldsymbol{x})$: This holds true since $v(0) = \boldsymbol{x}$, and our primary goal is to compute $F(\boldsymbol{x})$.
2) $\psi(0) = F(v(0)) \cdot w(0) = \alpha \cdot F(v(0)) = \alpha \cdot \phi(0)$: This equation is derived from the fact that $w(0) = \alpha$, which implies $\psi(0) = \alpha \cdot \phi(0)$.

If all CSNs compute accurately, $\psi(0) = \alpha \cdot \phi(0)$ must hold. If there is an erroneous CSN, $\hat{\psi}(0) \neq \alpha \cdot \hat{\phi}(0)$ unless α is known. As α is secret-shared, its exact value is hidden from malicious entities without enough shared points. Any deviation from $\hat{\psi}(0) = \alpha \cdot \hat{\phi}(0)$ is evidence of fraud, allowing MTD to verify the authenticity of the final computation results within the Meta-BMEOC framework.

B. Privacy

As previously defined, when the combined number of semi-honest CSNs and malicious CSNs is less than the threshold *t*, neither any CSN nor a third party can access MTD's computational tasks or the final computation result. Meta-BMEOC employs the Shamir's secret sharing scheme to distribute computational tasks among *m* CSNs, which then reconstruct the final computation result using interpolation. Thus, the privacy assurance of Meta-BMEOC can be expressed through the following theorem:

- *Theorem 1:* For any subset $\mathcal{S}$ of size *t*, where $\mathcal{S} \subseteq [m]$, and for any $F \in \mathcal{P}(q, k, d)$, the distributions of $\sigma(\mathcal{S}, F, \boldsymbol{x}^0)$ and $\sigma(\mathcal{S}, F, \boldsymbol{x}^1)$ are indistinguishable.
- *Proof:* Consider any subset $\mathcal{S}$ of size *t*, with $\mathcal{S} \subseteq [m]$, and any $F \in \mathcal{P}(q, k, d)$. For any $\boldsymbol{x}^0, \boldsymbol{x}^1 \in \mathbb{F}_q^k$, we observe:

$$\sigma\Big(\mathcal{S}, F, \boldsymbol{x}^0\Big) = \{(v(i), w(i)) | i \in \mathcal{S}\} \tag{16}$$

$$\sigma\Big(\mathcal{S}, F, \boldsymbol{x}^1\Big) = \{(v(i), w(i)) | i \in \mathcal{S}\}. \tag{17}$$

Here, $c(u) = \boldsymbol{x}^j + \mathrm{r}_1 u + \cdots + \mathrm{r}_t u^t$, with $j \in \{0, 1\}$. Given that the coefficients $\mathrm{r}_1, \ldots, \mathrm{r}_t$ are randomly chosen, the distribution of the secret shares each server receives remains the same irrespective of the value of $\boldsymbol{x}^j$. Therefore, both $\sigma(\mathcal{S}, F, \boldsymbol{x}^0)$ and $\sigma(\mathcal{S}, F, \boldsymbol{x}^1)$ have indistinguishable distributions. Consequently, an adversary, even with polynomial-time capabilities and under any attack scenario, cannot discern any information about the computational tasks or the final computation result through any meaningful statistical analysis of MTD outputs or their inputs. ■

C. Accountability

Within Meta-BMEOC, every outsourced computational result is digitally signed by the respective CSN using elliptic curve cryptography. This assures two pivotal security attributes:

- *Non-repudiation:* With each computational result accompanied by the CSN's signature, the CSN cannot later deny their previously submitted computation outcomes.
- *Integrity and Authenticity:* The MTD is unable to fabricate a computational result $\hat{R}_J = (ID_{T_J}, c_J, b_J, Sig_{T_J})$ to implicate the CSN, since the MTD cannot counterfeit the CSN's signature.

For every submitted result $R_J = (ID_T, c_J, b_J)$ and the associated task $T_J = (ID_T, f_J, \partial_J, pk_M, Sig_T)$, the Enclave

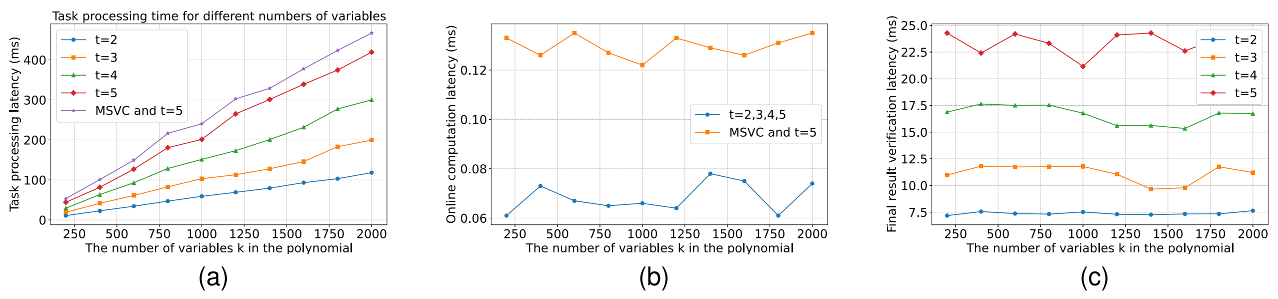


Fig. 4. Latency Analysis for Varying Numbers of Variables (k) in different stages: (a) Task Processing, (b) Online Computation, and (c) Final Result Verification.

repeats the computation of the outsourced task. When $c_J' \neq c_J$ or $b_J' \neq b_J$, CSN_J is identified as malicious. This step is precise, adeptly pinpointing CSNs that have submitted erroneous computational results, without false accusations. Upon detecting a malicious CSN, the Meta-BMEOC system not only recognizes it but can also implement financial penalties via blockchain and off-chain smart contracts, such as forfeiting their deposits.

From the above discourse, it is evident that under the ideal security modelassuming that the Enclave hardware is trustworthy and secureMeta-BMEOC showcases robust accountability; it can effectively identify and hold accountable those malicious CSNs that submit incorrect computational results.

VI. Experimental Evaluation

This section conducts a performance evaluation of the Meta-BMEOC framework, focusing on task processing, online computation, and the time cost for result verification and reconstruction. We compare Meta-BMEOC with the solution proposed by Zhang and Wang [2], specifically their second proposed solution. For public accountability, comparisons are drawn with the FVP-EOC edge outsourcing computing scheme by Li et al. [6], which also uses blockchain to identify malicious servers.

A. Experimental Setup

Our performance evaluation was conducted using a Raspberry Pi (simulating the MTD client) and a dedicated server (simulating the CSN server). The Raspberry Pi is equipped with a 1.5 GHz quad-core CPU and 4 GB RAM, running Ubuntu 18.04.4 LTS; the server has a 2.5 GHz quad-core CPU and 8 GB RAM, operating on Ubuntu 20.04.6 LTS. The security level is set to 128-bit ($\lambda = 128$, $q = 2^{128} + 51$). Finite field computations were implemented using the FLINT library (version 2.8.0), and both the outsourced computation protocol of this paper and the accountability algorithm for outsourced computations proposed by Li et al. [6] were written in C language.

Given that Meta-BMEOC operates on a permissioned consortium blockchain, the network implements stringent registration and auditing processes, permitting only authorized nodes to participate. As a result, the proportion of malicious nodes is generally low, and any node identified as malicious is promptly removed. Furthermore, a larger threshold t could potentially compromise the universality of outsourced computational tasks. In light of Meta-BMEOC's relatively secure environment, our experimental setup selected thresholds of $t = \{1, 2, 3, 4, 5\}$ to maintain an optimal balance between security and efficiency. For the polynomial variable number k, we considered $k \in \{200, 400, \ldots, 2000\}$. In terms of the polynomial degree d, we primarily focused on $d = 2$ while also evaluating outsourced computations for $d = \{4, 6, \ldots, 20\}$.

B. Implementation Details

In our implementation, we executed task processing and the final result verification on the Raspberry Pi, while online computations were performed on the server. First, we measured the time taken for task processing, online computation, and final result verification as the polynomial variable number increases, with $d = 2$ and varying values of t. These results are illustrated in Figures 4(a) to 4(c).

According to Figure 4(a), with the increment in the number of variables k, the duration for task processing in all methods shows a gradual increase. This trend is primarily attributed to the fact that the time complexity of fundamental polynomial operations, such as addition, multiplication, and evaluation, correlates with the variable number k. For instance, during multivariate polynomial multiplication, the highest time complexity reaches $O(k^2 \times \mathcal{N}^2)$, where $\mathcal{N} = \binom{k+d}{d}$ denotes the combinations of choosing d elements out of $k+d$, representing the polynomial's term count.

Simultaneously, augmenting t also impacts the time consumption of task processing. Within Shamir's Secret Sharing framework, as the threshold t increases, the polynomial necessitates a greater number of evaluation points. Consequently, the complexity of interpolation computations and the volume of linear equations to be addressed both experience a surge, resulting in an extended computational time for tasks.

The task processing procedure in Meta-BMEOC involves operations such as polynomial generation, Shamir's Secret Sharing, and signature creation. The computational complexity of these operations is $O(m \times t \times k + \log \ell)$, where ℓ denotes the length of the ECC key. As depicted in 4(a), Meta-BMEOC demonstrates a reduced computational time for task processing (specifically at $t = 5$, with analogous patterns observed for other t values) when juxtaposed with the method proposed by Zhang and Wang [2]. The predominant rationale behind this is that in an open edge environment characterized by zero-trust, Zhang et al. are compelled to employ digital signatures to affirm the integrity and authenticity of computational tasks. The frequency of digital signature operations directly corresponds with the number of participating edge servers in the computation, defined as $m = (d + 1)t + 1$. Conversely, courtesy of the incorporation of blockchain technology and off-chain smart contracts, Meta-BMEOC circumvents the repeated execution of digital signature operations. A singular digital signature suffices, allowing edge servers to subsequently ascertain their task hash values and authenticate the integrity and genuineness of messages using off-chain smart contracts. It is pivotal to recognize that Zhang et al. exclusively detailed the core algorithms in their paper without addressing the integrity and veracity of message transmission, thereby omitting the digital signature component. Nonetheless, in a zero-trust open-edge milieu, the indispensability of digital signatures is a foregone conclusion.

The online computation process in Meta-BMEOC involves operations such as digital signature verification, task computation, and signature generation. The computational complexity of these operations is $O(\binom{k+d}{d} \times k + \log \ell)$. According to Figure 4(b), changes in the security threshold t do not affect the online computation time of edge servers; this is because, for security and privacy considerations, Meta-BMEOC mandates that each edge server can only obtain a single secret share for computation. It is worth noting that Figure 4(b) displays the maximum online computation time for an individual edge server. Analogous to Figure 4(a), when applying the scheme of Zhang and Wang [2] in a zero-trust open edge environment, it is necessary to preliminarily verify the digital signature of the client to ensure the correctness and integrity of the computational task. Consequently, their online computation time is marginally longer.

The result verification and reconstruction process in Meta-BMEOC involves operations such as digital signature verification and polynomial interpolation. The computational complexity of these operations is $O(m \log \ell + m^2)$. As indicated in Figure 4(c), the number of polynomial variables k has a minimal impact on the time the client takes to verify the final computation results. However, the privacy and security threshold t does influence the verification time for the final results; this is due to a larger t necessitating more interpolation terms and linear equations to confirm the correctness of the final computation results.

C. Evaluation of Accountability Mechanisms

As depicted in Figure 5, concerning accountability for malicious edge servers, we compared the accountability capabilities of Meta-BMEOC with FVP-EOC [6]. Essentially, FVP-EOC ensures the correctness of outsourced computation results through redundant computation and a voting mechanism. As such, to ensure that malicious edge servers cannot dictate the final outcome, the number of honest edge servers should significantly outnumber the malicious ones. Specifically, FVP-EOC verifies the correctness of the computation results by comparing the similarity of ElGamal

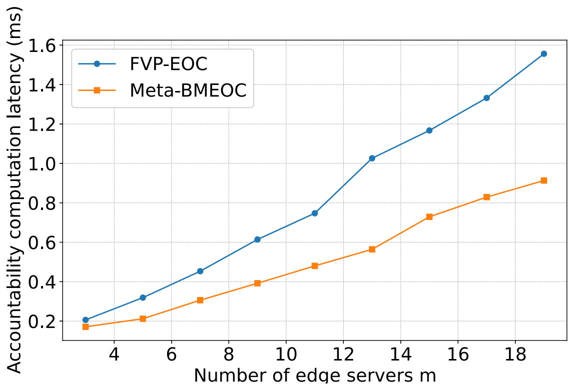


Fig. 5. Latency of Accountability Computation for Varying Numbers of Servers (m).

ciphertexts returned from different edge servers. To circumvent cryptographic computations and the need for more servers, Meta-BMEOC employs off-chain smart contracts to carry out redundant computations within enclaves, thereby identifying and holding malicious servers accountable. It's important to note that Meta-BMEOC only requires redundant computations within enclaves if the verification of the final computation result fails, as the verification of the final computation result and the identification and accountability of malicious servers are decoupled in Meta-BMEOC. Interestingly, without utilizing off-chain smart contracts, Meta-BMEOC can seamlessly adapt to the accountability mechanism of FVP-EOC. However, this would necessitate at least three times the number of edge servers. Given that this imposes a higher requirement on the number of servers available in the edge environment, we did not opt for the Meta-BMEOC mechanism.

D. Evaluation of High-Degree Polynomial

In this section, we assess the computational tasks associated with high-degree polynomials. Analogous to the low-order polynomial case, we perform task processing and final result verification on a Raspberry Pi and conduct online computations on the server. However, in the context of high-degree polynomials, the number of terms, represented by $\mathcal{T} = \binom{k+d}{d}$, escalates rapidly with increasing d. For instance, when $d = 20$ and $k = 20$, the polynomial term counts $\mathcal{T} = 137846528820$, surpassing the Raspberry Pi's memory capabilities. To better capture the computational time, we opted for a reduced k value, specifically $k = 9$.

Figure 6(a) illustrates that as the polynomial's degree d increases, there is a marked increase in the computational overhead for task processing. This is attributed to the relationship $m = (d + 1)t + 1$. With t held constant, a rise in d necessitates an expanded server count m. Consequently, the client is tasked with generating an increased number of secret share portions, denoted as $\sigma_\jmath = (c(\jmath), b(\jmath))$, elevating the client's computational load. According to Figure 6(b), the polynomial's degree d exerts a negligible impact on the server's online computation time, given the server's ample processing capacity tailored for high-degree polynomial computations. As highlighted in Figure 6(c), the polynomial degree d distinctly influences the

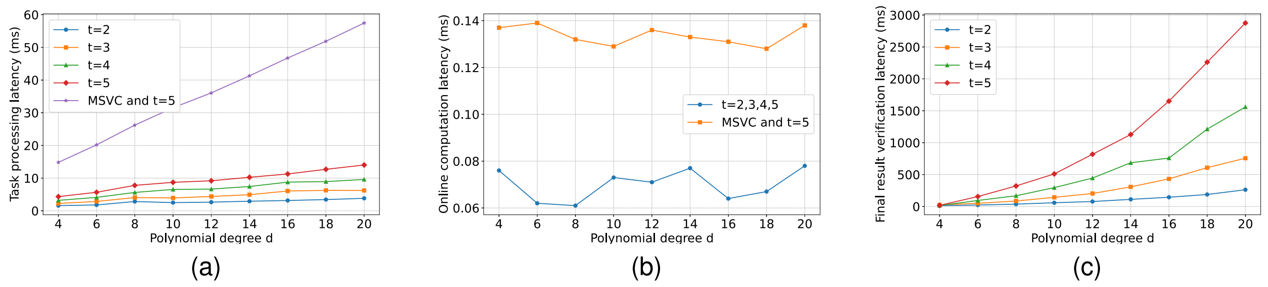


Fig. 6. Latency Analysis for Varying Polynomial Degrees (d) in different stages: (a) Task Processing, (b) Online Computation, and (c) Final Result Verification.

server's final result verification time, this is rooted in the fact that an elevated d not only amplifies the client's interpolation terms and requisite linear equations but also extends the quantity of server-returned signatures the client must validate.

VII. Conclusion

This paper introduces the Meta-BMEOC framework, a blockchain-based mobile edge outsourcing computing framework specifically designed for the Metaverse. It aims to provide low-latency edge outsourcing computing services that ensure privacy protection and accountability. The framework consists of a blockchain service layer, an off-chain smart contract layer, and a computing service layer, which establishes a trust relationship between Mobile Terminal Devices (MTD) and mobile edge servers through its blockchain service layer. Meta-BMEOC employs a privacy-preserving outsourced computing protocol that allows MTDs to outsource tasks via a threshold secret sharing protocol and locally verify the results. Additionally, the framework prevents collusion among malicious servers through TEE-supported off-chain smart contracts and holds them accountable in the event of computational errors. We have also designed an incentive mechanism to enhance the system's security and sustainability. One of the key advantages of Meta-BMEOC is its low-latency performance and reliance on non-complex cryptographic outsourced computing services. Future expansions will aim to support a broader range of computational tasks, further enhancing performance and security.

References

[1] Y. Fu, C. Li, F. R. Yu, T. H. Luan, P. Zhao, and S. Liu, "A survey of blockchain and intelligent networking for the metaverse," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3587–3610, Feb. 2023.

[2] L. F. Zhang and H. Wang, "Multi-server verifiable computation of low-degree polynomials," in *Proc. IEEE Symp. Security Privacy (SP)*, 2022, pp. 596–613.

[3] L. Wang, Y. Tian, and J. Xiong, "Achieving reliable and anti-collusive outsourcing computation and verification based on blockchain in 5G-enabled IoT," *Digit. Commun. Netw.*, vol. 8, no. 5, pp. 644–653, 2022.

[4] X. Yu, Z. Yan, and R. Zhang, "Verifiable outsourced computation over encrypted data," *Inf. Sci.*, vol. 479, pp. 372–385, Apr. 2019.

[5] D. Li, J. Wu, J. Le, X. Liao, and T. Xiang, "A novel privacy-preserving location-based services search scheme in outsourced cloud," *IEEE Trans. Cloud Comput.*, vol. 11, no. 1, pp. 457–469, Jan.–Mar. 2023.

[6] T. Li, Y. Tian, J. Xiong, and M. Z. A. Bhuiyan, "FVP-EOC: Fair, verifiable, and privacy-preserving edge outsourcing computing in 5G-enabled IIoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 940–950, Jan. 2023.

[7] H. Long, C. Xu, G. Zheng, and Y. Sheng, "Socially-aware energy-efficient task partial offloading in MEC networks with D2D collaboration," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 3, pp. 1889–1902, Sep. 2022.

[8] R. Lai and G. Zhao, "Blockchain for achieving accountable outsourcing computations in edge computing," *Comput. Commun.*, vol. 200, pp. 17–29, Feb. 2023.

[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," in *Proc. Decentralized Bus. Rev.*, 2008, Art. no. 21260.

[10] T. Frassetto et al., "POSE: Practical off-chain smart contract execution," in *Proc. 30th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2023, pp. 1–20.

[11] M. Rivinius, P. Reisert, D. Rausch, and R. Küsters, "Publicly accountable robust multi-party computation," in *Proc. IEEE Symp. Security Privacy (SP)*, 2022, pp. 2430–2449.

[12] H. Qushtom, J. Mišić, V. B. Mišić, and X. Chang, "Efficient blockchain scheme for IoT data storage and manipulation in smart city environment," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 3, pp. 1660–1670, Sep. 2022.

[13] J. Li, D. Li, and X. Zhang, "A secure blockchain-assisted access control scheme for smart healthcare system in fog computing," *IEEE Internet Things J.*, vol. 10, no. 18, pp. 15980–15989, Sep. 2023.

[14] S. H. Alsamhi et al., "Drones' edge intelligence over smart environments in B5G: Blockchain and federated learning synergy," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 295–312, Mar. 2022.

[15] Z. Zhou et al., "Blockchain-based secure and efficient secret image sharing with outsourcing computation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 23, no. 1, pp. 423–435, Jan. 2024.

[16] P. Razmjouei, A. Kavousi-Fard, M. Dabbaghjamanesh, T. Jin, and W. Su, "DAG-based smart contract for dynamic 6G wireless EVs charging system," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 3, pp. 1459–1467, Sep. 2022.

[17] Z. Wan, Y. Zhou, and K. Ren, "Zk-AuthFeed: Protecting data feed to smart contracts with authenticated zero knowledge proof," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1335–1347, Mar./Apr. 2023.

[18] W. Zhao et al., "Privacy-preserving outsourcing of K-means clustering for cloud-device collaborative computing in space-air-ground integrated IoT," *IEEE Internet Things J.*, vol. 10, no. 23, pp. 20396–20407, Dec. 2023.

[19] X. Liu, Y. Zheng, X. Yuan, and X. Yi, "Securely outsourcing neural network inference to the cloud with lightweight techniques," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 620–636, Jan./Feb. 2023.

[20] Y. Guan, H. Zheng, J. Shao, R. Lu, and G. Wei, "Fair outsourcing polynomial computation based on the blockchain," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2795–2808, Sep./Oct. 2022.

[21] L. Guo, Q. Wang, and W.-C. Yau, "Online/offline rewritable blockchain with auditable outsourced computation," *IEEE Trans. Cloud Comput.*, vol. 11, no. 1, pp. 499–514, Jan.–Mar. 2023.

[22] W. Susilo, F. Guo, Z. Zhao, Y. Jiang, and C. Ge, "Secure replication-based outsourced computation using smart contracts," *IEEE Trans. Services Comput.*, vol. 16, no. 5, pp. 3711–3722, Sep./Oct. 2023.

[23] Z. Hou, J. Ning, X. Huang, S. Xu, and L. Y. Zhang, "Blockchain-based efficient verifiable outsourced attribute-based encryption in cloud," *Comput. Stand. Interfaces*, vol. 90, Jul. 2024, Art. no. 103854.

[24] H. Wang et al., "A publicly verifiable outsourcing matrix computation scheme based on smart contracts," *IEEE Trans. Cloud Comput.*, vol. 12, no. 1, pp. 70–83, Jan.–Mar. 2024.

[25] J. Liu and L. F. Zhang, "Privacy-preserving and publicly verifiable matrix multiplication," *IEEE Trans. Services Comput.*, vol. 16, no. 3, pp. 2059–2071, May/Jun. 2023.
[26] Z. Liu et al., "A privacy-preserving outsourcing computing scheme based on secure trusted environment," *IEEE Trans. Cloud Comput.*, vol. 11, no. 3, pp. 2325–2336, Jul.–Sep. 2023.
[27] B. Qu, Y. Bai, Y. Chu, L.-e. Wang, F. Yu, and X. Li, "Resource allocation for MEC system with multi-users resource competition based on deep reinforcement learning approach," *Comput. Netw.*, vol. 215, Oct. 2022, Art. no. 109181.
[28] Z. Zhang et al., "QKPT: Securing your private keys in cloud with performance, scalability and transparency," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 478–491, Jan./Feb. 2023.
[29] F. Wang, J. Mickens, N. Zeldovich, and V. Vaikuntanathan, "Sieve: Cryptographically enforced access control for user data in untrusted clouds," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2016, pp. 611–626.
[30] C. Li, W. Qiu, X. Li, C. Liu, and Z. Zheng, "A dynamic adaptive framework for practical Byzantine fault tolerance consensus protocol in the Internet of Things," *IEEE Trans. Comput.*, vol. 73, no. 7, pp. 1669–1682, Jul. 2024.
[31] A. Muñoz, R. Rios, R. Román, and J. López, "A survey on the (in) security of trusted execution environments," *Comput. Secur.*, vol. 129, Jun. 2023, Art. no. 103180.
[32] Y. Li, D. Zeng, L. Gu, A. Zhu, Q. Chen, and S. Yu, "PASTO: Enabling secure and efficient task offloading in TrustZone-enabled edge clouds," *IEEE Trans. Veh. Technol.*, vol. 72, no. 6, pp. 8234–8238, Jun. 2023.
[33] D. C. G. Valadares, N. C. Will, M. A. Spohn, D. F. de S. Santos, A. Perkusich, and K. C. Gorgonio, "Confidential computing in cloud/fog-based internet of things scenarios," *Internet Things*, vol. 19, Aug. 2022, Art. no. 100543.
[34] A. Haeberlen, P. Kouznetsov, and P. Druschel, "The case for Byzantine fault detection," in *Proc. HotDep*, 2006, pp. 1–6.

Chunpei Li received the Ph.D. degree from the School of Computer Science and Engineering, Guangxi Normal University in 2024, where he is currently conducting postdoctoral research with the Ministry of Education Key Laboratory of Educational Blockchain and Intelligent Technology. His research interests include blockchain, artificial intelligence, and information security.

Chen Liu received the Ph.D. degree from the School of Computer Science and Engineering, Beihang University, China, in 2024. She is currently a Research Assistant with Zhongguancun Laboratory. Her research interests include machine learning, cyber security, and data mining.

Peng Liu received the Ph.D. degree from Beihang University, China, in 2017. He joined Guangxi Normal University as an Assistant Professor in 2007. Since 2015, he has been an Associate Professor. His current research interests include federated learning and blockchain.

Xianxian Li received the Ph.D. degree from the School of Computer Science and Engineering, Beihang University, Beijing, China, in 2002, where he worked as a Professor from 2003 to 2010. He is currently a Professor with the School of Computer Science and Information Technology, Guangxi Normal University, Guilin, China. His research interest includes information security, blockchain, and privacy computing.

Wangjie Qiu received the Ph.D. degree from Beihang University, China, in 2012. He is currently an Associate Research Fellow with the Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing. He is an essential open-source contributor to the blockchain platform-ChainMaker. His research interests include cryptography, blockchain, and privacy computing.

Lei Lei received the master's degree from the School of Electric Engineering, Guangxi University, Nanning, China, in 2014. She is currently pursuing the Doctoral degree with the School of Computer Science and Information Technology, Guangxi Normal University, Guilin, China. She research interest includes blockchain and machine learning.

Yanli Jin is currently pursuing the master's degree with the School of Computer Science and Engineering, Guangxi Normal University, Guilin, China. She research interest is blockchain.