

# A Blockchain-Based Audit Mechanism for Trust and Integrity in IoT-Fog Environments

Ismael Martinez, Abdelhakim Senhaji Hafid, Michel Gendreau

**Abstract**—The full realization of smart city technology is dependent on the secure and honest collaboration between IoT applications and edge-computing. In particular, resource constrained IoT devices may rely on fog-computing to alleviate the computing load of IoT tasks. Mutual authentication is needed between IoT and fog to preserve IoT data security, and monetization of fog services is needed to promote the fog service ecosystem. However, there is no guarantee that fog nodes will always respond to IoT requests correctly, either intentionally or accidentally. In the public decentralized IoT-fog environment, it is crucial to enforce integrity among fog nodes. In this paper, we propose a blockchain-based system that 1) streamlines the mutual authentication service monetization between IoT and fog, 2) verifies the integrity of fog nodes via service audits, and 3) discourages malicious activity and promotes honesty among fog nodes through incentives and penalties.

**Index terms**— Internet of Things, fog computing, blockchain, service auditing, mutual authentication, smart contracts, zero-knowledge proof of membership

## I. INTRODUCTION

THE *Internet of Things* (IoT) is an ever growing paradigm of sensors and computing devices inter-connected through the internet. IoT has emerged in both public and private sectors with the main objective of facilitating our lives [1]. A wide scale network of collaborative IoT applications is the first step towards the implementation of smart cities [2].

Many IoT devices and applications rely on external computation and storage due to limited internal resources. Though Cloud data-centers are heavily equipped to support any number of IoT requests, network congestion near distant Cloud data-centers may result in high response latency to IoT devices [3]. This high latency can be an inhibiting factor for certain real-time IoT applications in health care [4], autonomous vehicles [5], and multimedia [6].

*Fog-computing* is a computational extension of Cloud services to the edge of the network. The fog layer is composed of geographically distributed ‘micro data-centers’, or nodes, that are positioned to support IoT with minimal latency [7]. Indeed, fog, alongside IoT and Cloud, are integral in creating an energy-efficient network computing architecture for smart cities [8], [9], [10].

I. Martinez is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, Canada H3C 3J7 (e-mail: ismael.martinez@umontreal.ca).

A. S. Hafid is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, Canada H3C 3J7 (e-mail: ahafid@iro.umontreal.ca).

M. Gendreau is with the Department of Mathematics and Industrial Engineering, Polytechnique Montreal, Quebec, H3C 3A7, Canada (email:michel.gendreau@polymtl.ca)

Current research in fog-computing focuses on the effective design of fog infrastructures [10], and resource off-loading policies from IoT to fog nodes [11]. However, such research does not consider the mutual needs of IoT and fog. IoT devices require real-time, secure and correct service from an authenticated server. Fog nodes require payment and advertisement for services from authenticated sources.

Furthermore, current work assumes that fog nodes always behave with *integrity*. That is, IoT devices are meant to blindly trust fog nodes even though it is possible that a fog node returns a faulty response, either intentionally or accidentally [12]. Indeed, the IoT-fog environment is **trustless** and currently lacks accountability for fog nodes to behave correctly. If IoT devices rely on fog nodes for computational processing, it is critical that we ensure active fog nodes are processing correctly, and eject malicious fog nodes from the IoT-fog environment.

In addition, IoT networks can be easy to tamper with and compromise without proper security measures. Blockchain technologies have been studied as a possible solution to provide security, privacy and access control to IoT due to the decentralization, immutability and high transparency of blockchain [13]. Hence, blockchain can be used to provide a secure line of communication between IoT and fog via mutual authentication [14]. Furthermore, blockchain can streamline the payment process from IoT for fog computing services, and enforce integrity among the fog nodes.

Based on observations of IoT-fog requirements, and limitations of current work, there exists a need for a single streamlined system in a trustless IoT-fog environment that 1) mutually authenticates IoT and fog prior to service, 2) facilitates service payment from IoT to fog, 3) verifies and holds malicious fog nodes accountable, and 4) benefits honesty and discourages malicious activity among fog nodes.

Inspired by current data auditing techniques [15], we propose a service auditing process for fog-computing to enforce computational integrity. To the best of our knowledge, this is the first attempt to enforce the service integrity of fog via a service auditing scheme. We also integrate current mutual authentication [14], [16] and fog monetization schemes [17], [18] into a single blockchain application, and leverage blockchain-enabled fog nodes to decrease latency [19]. That is, we propose the *Fog Identity & Service Integrity Enforcement* (FISIE) system that streamlines IoT-fog authentication, service, monetization, and integrity auditing through a single smart contract. The FISIE smart contract described in this paper is a Proof-of-Concept based on Ethereum [20]. However, any other smart-contract capable blockchain platform would be compatible

with this system.

Our contributions are as follows:

- We review and summarize current literature related to payment, service and mutual authentication.
- We propose a general architecture of heterogeneous IoT and blockchain-enabled fog that is compatible with any smart contract-enabled blockchain.
- We define a smart contract-based system for mutual authentication, monetization, and service auditing.
- We describe a penalty system to enforce service integrity.
- We discuss the security of the system, and analyze various auditing scheduling policies for optimal system integrity.

The remainder of this paper is organized as follows. Section II reviews the current contributions in related fields to inspire our solution. Section III provides an overview of the different components of the FISIE system. Section IV provides background knowledge and configurations specifics of blockchain, cryptography, and the IoT-fog physical layers. Section V initializes the smart contract. Sections VI, VII and VIII respectively define the identity management, payment management and integrity verification functions of the smart contract. Section IX describes how the smart contract functions provide penalties and incentives for fog integrity. Section X discusses the FISIE system security, and analyses the affects of different sampling policies on long-term fog integrity. Finally, section XI summarizes future work and concludes the paper.

## II. RELATED WORK

We are interested in providing security and integrity to the IoT-fog environment without significantly increasing communication latency. Our reviewed literature focuses on the state-of-the-art in a) IoT-fog security, b) data auditing of fog, c) blockchain-based monetization, and d) blockchain-fog integration.

### A. IoT-fog security

Two of the key elements in providing security to any system is the inclusion of authorization & authentication [13]. In particular for the IoT-fog environment, we review implementations of access control for IoT data, and mutual authentication between IoT and fog.

1) *Authorization*: An *access control* policy defines which entities have the authority to access the data of which devices. Access control policies may list individual valid entities, or list attributes that entities must have to gain access [21]. A micro server such as fog has sufficient storage and computing resources to define and validate its own access control policy. However, IoT devices have minimal resources, and may not be able to store its own list of valid entities. In this case, the IoT access control policies are stored in a separate trusted server with sufficient resources.

Algarni et al. [22] propose a blockchain-based access control scheme for IoT. This scheme takes advantage of the transparency and security of blockchain to house all IoT access control policies. Since the blockchain itself cannot be hosted on the IoT devices, the fog layer can be used to host the blockchain and decrease communication latency between the blockchain and IoT.

2) *Authentication*: The authorization process often works in tandem with an *authentication* mechanism to prove the identity of a communicating entity [21]. The authentication process is crucial in protecting IoT and fog from security risks such as man-in-the-middle attacks and replay attacks [23]. Secure authentication in IoT, fog and Cloud are often based in standard encryption schemes such as RSA or elliptic curve cryptography (ECC), though ECC is known to be more secure than RSA for equivalent key sizes [24]. In the IoT-fog environment, we are interested in *mutual authentication*, wherein an IoT device and a fog node authenticate each other prior to communicating and data sharing [25].

Singh and Chaurasiya [25] propose a lightweight mutual authentication scheme with a centralized Cloud data-center as a trusted third-party. Based on ECC, the Cloud data-center sets all relevant cryptographic parameters, while IoT devices and fog nodes store only their own public keys. That is, private keys are stored on Cloud instead of the IoT/fog devices. Though this scheme is lightweight, storing minimal data on IoT devices, it requires absolute trust in the Cloud data-center. In addition, requiring communication with the Cloud increases communication latency for IoT.

Instead, we consider the use of a decentralized blockchain for the authentication process. Current contributions [14], [16], [26] use a smart contract-based scheme to register or remove identification information from IoT devices and fog nodes. Once registered, the information is stored and queried from a trusted off-chain table. However, these schemes rely on an additional centralized registration authority to generate and store keys for IoT devices and fog nodes [14], [16]. Giving a centralized authority this level of control over the system's private keys is a potential security risk. Instead, we propose to limit the use of any off-chain resources, and keep all private keys on their respective devices.

Patwary et al. [26] propose a blockchain-based mutual authentication scheme that uses the physical fog location data as part of the authentication process. Though the use of centralized resources are limited, this scheme only works with stationary IoT devices and fog nodes since authentication relies on a static location validation. Instead, we seek to implement a generalized authentication scheme that allows for device mobility without compromising IoT-fog security.

### B. Data auditing of Fog

Several contributions propose a similar data auditing scheme to verify the data replica cache of edge servers [12], [15], [27]. A vendor who has previously cached its own data to edge servers may request the hash of the data replica from edge servers. The vendor compares the hash with its own data hash to verify the data integrity of an edge server.

Zikratov et al. [28] propose a data auditing scheme based on a private blockchain. Data is distributed to clients and is also stored on the blockchain. Periodically, the client data is downloaded and verified with the blockchain data by a third party auditor.

Tian et al. [29] address the problem of data auditing in a public IoT-fog environment. They propose to tag IoT data

which is sent to a fog node which places its own tag, and then sends it to the Cloud. A third party auditor can then verify the integrity of the fog nodes via a zero-knowledge proof of integrity.

In both cases [28], [29], absolute cooperation is needed from fog nodes to honestly share or allow access to its server data. This level of trust cannot be guaranteed in a trustless system.

### C. Blockchain-based Monetization

Service payments from IoT to fog have been previously considered by means of blockchain smart contracts [17], [18]. Debe et al. [17] consider a monetization smart contract in which IoT devices deposit Ether, which are then used to pay for fog services. Huang et al. [18] use a smart contract to hold a collateral deposit from IoT until the IoT device directly pays the fog node. If payment is not processed in a timely manner, the collateral is given to the fog node.

The system by Huang et al. [18] uses a commitment-based sampling approach in which the IoT devices samples a portion of the result from the fog node, to decide whether to pay or not. In such a case that the IoT device is not satisfied with the fog results and decides not to pay, it may start a dispute with a third party to retrieve its deposit.

Note, that this proposed system [18] requires a separate blockchain transaction for the 1) initial deposit, 2) a confirmation of deposit from fog, 3) sending a separate payment from IoT to fog, 4) then returning the deposit to IoT. This payment process can be costly in blockchain fees due to the total number of required transactions. Furthermore, this process requires verification of the result from the IoT device, which may not be computationally possible from resource-constrained devices.

### D. Blockchain-fog integration

Fog nodes are geographically distributed, and blockchains are replicated and hosted on distributed servers. Therefore, it is reasonable to combine these concepts to minimize the communication latency between fog nodes and the blockchain. *Blockchain-enabled fog nodes* are fog nodes that use a portion of their resources to host a copy of the blockchain. By doing so, all communication delay between fog and blockchain is eliminated. Almadhoun et al. [23] uses blockchain-enabled fog nodes for IoT authentication – a process whose speed is highly dependent on communication delay between IoT, fog and blockchain. The resource requirements of the blockchain can be further reduced by using ‘light nodes’ which use block summarization to reduce the amount of data stored on the fog node [19], [30], [31]. In particular, in this paper we store only blockchain data relevant to the authentication and auditing processes.

### E. Summary of Reviewed Literature

None of the reviewed contributions consider a penalty or action to be taken if a fog node or edge server is found to be corrupted. If the corruption is accidental, then the appropriate server could be given the correct data. However,

if the corrupted data is malicious, then there is no penalty to stop the server from continuing to alter cached or processed IoT data. At worst, a fog node that returns malicious data is simply not paid [18].

In public systems, a call-and-response process of requesting a proof of data integrity from servers must be taken. However, there is no incentive given for the data servers to comply with the audit [12], [15], [27]. To validate service integrity, it is left to the IoT device to verify the work done by the fog is correct before giving payment, a task which is not always computationally feasible by resource constrained IoT [18]. Even in private networks [28], it may not be reasonable to have a third party auditor with full accessibility of client files without major privacy concerns of individuals.

The focus of this paper is to validate the service integrity of fog nodes who are meant to support IoT. We take inspiration from related work to form the FISIE system, a blockchain-based system that streamlines IoT-fog mutual authentication, fog service monetization, and verification of fog integrity. We also recognize the benefits of integrating blockchain within the fog layer for low-latency mutual authentication. We find the addition of an incentive and penalty mechanism to be necessary to enforce auditing cooperation from fog nodes and overall IoT-fog system integrity. A comparison of the proposed FISIE system with other contributions is shown in Table I.

## III. FISIE SYSTEM OVERVIEW

The FISIE system aims to 1) streamline IoT-fog mutual authentication and fog service monetization, 2) verify the integrity of fog nodes via external service auditing, and 3) promote the honest collaboration between IoT and fog via incentives and penalties. These objectives are accomplished via the *Identity & Integrity Management Smart Contract* (IIMSC) which interacts with IoT, fog, and an *oracle* – an off-chain semi-trusted third-party. A generalized blockchain structure will increase the likelihood by others of adopting the FISIE system. Hence, though our default implementation uses Ethereum [20], other implementations may use smart contract-capable blockchain platforms such as Solana<sup>1</sup> or Layer 2 platforms such as Arbitrum<sup>2</sup> or Optimism<sup>3</sup> for scalability and lower processing fees [32], [33].

The key processes of IIMSC are summarized as 1) Identity Management, 2) Payment Management, and 3) Integrity Verification. The key processes of IIMSC are shown in Fig. 1. These key components are summarized below, and further explored in sections VI, VII, and VIII. These three components offer incentives and penalties for fog nodes to behave honestly (see Section IX).

### A. Identity Management

IIMSC defines *lookup tables* that hold information about IoT and fog blockchain addresses, current token holdings, and fog reputation. These lookup tables are used for mutual

<sup>1</sup><https://solana.com/>

<sup>2</sup><https://arbitrum.io/>

<sup>3</sup><https://www.optimism.io/>

<sup>4</sup>IoT icons by <https://www.avsystem.com>

Contribution	Mutual Authentication	Fog Service Monetization	Resource Constrained IoT-Compatible	Immutable (Blockchain)	Fog Penalization	Fog Honesty Verification	Public Fog Auditing	Penalization for Malicious Fog
Debe [17]	✗	✓	✓	✓	✗	✗	✗	✗
Almadhoun [23]	✓	✗	✓	✓	✗	✗	✗	✗
Singh [25]	✓	✗	✓	✗	✗	✗	✗	✗
Patwary [26]	✓	✗	✓	✓	✗	✗	✗	✗
Tian [29]	✗	✗	✗	✗	✗	✓	✓	✗
Huang [18]	✗	✓	✗	✓	✗	✓	✗	✗
FISIE	✓	✓	✓	✓	✓	✓	✓	✓

Table I: Inspiration for the FISIE system is taken from various contributions.

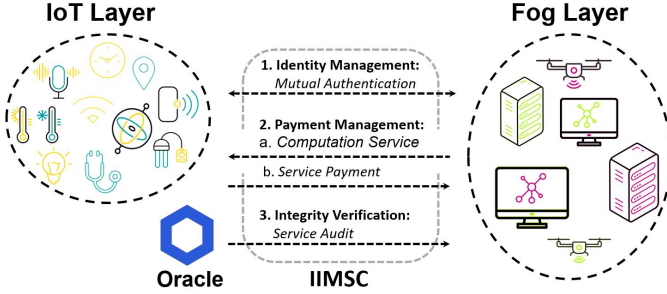


Figure 1: The main objectives of the FISIE system is to use a blockchain smart contract to 1) facilitate authentication and service and payment between IoT devices<sup>4</sup> and fog nodes, and 2) enforce service integrity among fog nodes.

authentication between IoT devices and fog nodes. Prior to participating in the FISIE system, all IoT devices and fog nodes must register with IIMSC. The blockchain addresses are used both to validate an entity's identity, as well as to forward payment. Current IoT token holdings are listed to ensure IoT devices have enough funds to pay for fog services. Fog devices have two fields for current token holdings – one for a collateral deposit to use the FISIE system, and another to accrue fog service payments. Both collateral deposit and a fog's reputation score give indication of a fog node's past behaviour, whether honest or malicious, i.e., their *reliability*. Hence, IoT devices may filter candidate fog nodes to which it is comfortable sending data based on a fog's reliability.

#### B. Payment Management

Once an IoT-fog pair has authenticated each other, an IoT device may send its request to the fog node for service. This is done off-chain via ECC encryption [34]. Once service is complete and successfully returned to the IoT device, a pre-determined amount is transferred from the IoT device's funds to the fog node's service payment funds via IIMSC lookup tables. That is, all funds remain within IIMSC until the fog node withdraws them.

#### C. Integrity Verification

To the best of our knowledge, there does not exist any research contributions in the enforcement of fog service integrity. We seek to implement a fog service auditing mechanism based on a call-and-response for the service output from

fog nodes for an IoT request. Unlike previous contributions, we include both an incentive to encourage fog nodes to participate in the call-and-response, and a penalty in the case the fog node fails the service audit.

We assume that IoT devices have limited resources and are therefore unable to verify the correctness of fog node's service. Therefore, we define a service auditing scheme to allow an oracle to verify the integrity of fog nodes without revealing its identity. Indeed, if a fog node was aware that it was being audited by the oracle, it may change its behaviour. Instead, by using a Zero-Knowledge Proof of Membership [35], the oracle disguises itself as another IoT device, encouraging the fog node to behave as it normally does.

#### D. Penalty & Incentive Mechanisms

The existence of an auditor in itself acts as a deterrent to malicious behaviour from fog nodes. If a fog node is found to return a faulty response, we employ a *penalty* mechanism to both reduce the fog node's collateral deposit and reputation score. If the audit is successful, the fog node's reputation score may increase up to a fixed cap. A higher reputation score may lead to more service requests from IoT, and hence more service payments. Therefore, such a reward also acts as an *incentive* for fog nodes to behave honestly.

### IV. DOMAIN BACKGROUND & CONFIGURATIONS

In this section, we provide background knowledge of blockchain, the IoT-fog environment, and cryptography, and define their role in the FISIE system.

#### A. Blockchain

Public blockchains, such as Bitcoin and Ethereum, are decentralized ledgers that enforce block consensus across all immutable blockchain nodes, which allows them to operate in a trustless environment [36]. Public blockchains also provide pseudo-anonymity, fault tolerance, and auditability. Blockchain technology is viewed as a key technology in adding security & privacy to IoT and industrial IoT (IIoT) applications [37].

1) *Smart Contracts*: A *smart contract* is an agreement between two or more parties that self-executes when specific conditions are met [38]. A smart contract on the blockchain benefits from the same immutability, persistency and auditability as other blockchain transactions. A common use case for

blockchain smart contracts is providing access control of IoT data, which can mitigate security & privacy issues in IoT [39]. The FISIE system's implementation of the *Identity & Integrity Management Smart Contract* (IIMSC) is compatible with any smart contract-capable blockchain.

2) *Oracles*: Often, a smart contract is dependent on real-world data to determine when its execution conditions are satisfied. However, the blockchain is isolated from the real-world internet environment, creating a need for a separate entity to convey the appropriate external information to the blockchain. An *oracle* is an off-chain third-party that is used to inject external data into smart contracts. Since oracles operate off-chain, it is necessary to validate the trustworthiness of both the oracle and the external data sources [40]. For this reason, centralized oracles are not often used since the validity of the communicated data from a single centralized entity cannot be trusted. Instead, multiple decentralized oracles are often used to cross-verify each other and create a trusted data feed [41]. The FISIE system relies on external decentralized oracles to audit fog nodes and trigger the appropriate smart contracts.

### B. Elliptic Curve Cryptography – Definitions & Settings

The FISIE system uses *elliptic curve cryptography* (ECC), a public-key cryptographic method that uses a globally agreed upon elliptic curve and base point over a finite field to generate public and private keys [34].

Consider the finite field  $\mathbb{F}_p$  for large prime number  $p$ . Over the elliptic curve  $E$ , beginning from a base point  $G$ , we select a secret key  $k$  and derive the public key  $P$  as

$$P = k \cdot G \quad (1)$$

where we ‘add’  $(\cdot) G$   $k$  times over the finite field of  $E$ .

1) *The elliptic curve*: Bitcoin and Ethereum use the `secp256k1` system for the Elliptic Curve Digital Signature Algorithm (ECDSA) [42] to sign blockchain transactions. The `secp256k1` elliptic curve is defined as

$$E : y^2 = x^3 + 7, \quad (2)$$

and produces 256-bit keys [43], [44]. Unlike other contributions that trust a third party to define the elliptic curve parameters and generate device keys [14], [25], we will simplify the process and use the built-in ECC parameters of these blockchains for secure IoT-fog communication.

2) *Encryption*: Suppose entity  $A$  has public key  $P_A$  and private key  $k_A$ , and entity  $B$  has public key  $P_B$  and private key  $k_B$ . Then a symmetric key can be formed between  $A$  and  $B$  since,

$$k_B P_A = k_B \cdot (k_A G) = k_A \cdot (k_B G) = k_A P_B. \quad (3)$$

Using this symmetric key, we can encrypt and decrypt a message between  $A$  and  $B$  using a symmetric encryption algorithm such as AES [34], [45].

3) *ECDSA*: ECDSA is the primary signature generation and verification algorithm in Bitcoin and Ethereum blockchains [44]. No transaction is accepted by the blockchain without a valid signature. Suppose a user  $A$  submits a signature to a verifier  $B$ . ECDSA enables a verifier  $B$  to recover the

public key  $P_A$  from a valid signature  $s$ . Hence, there is no need for  $A$  to submit  $P_A$  to  $B$ . For the remainder of this paper, every signature used is an ECDSA signature.

4) *One-way hash function*: We define the function  $H : \{0,1\}^* \mapsto \{0,1\}^{256}$  as a secure, one-way 256-bit *hash* function. Examples of viable hash functions with this property are SHA-256 and Keccak-256 used by Bitcoin and Ethereum ECDSA respectively [36], [20]. Importantly, these hash functions also derive a user's address by hashing the user's public key. The address of a user with public key  $P$  is defined as the last 20 bytes of the hash  $H(P)$  [20]. We define the operation  $||_n$  to be the  $n$ -byte right hand truncation of a value. Hence, a user with public key  $P$  has address  $H(P)||_{20}$ .

### C. Physical Architecture

Our proposed architecture is meant to be as general as possible so it may fit any existing IoT-fog infrastructure. Both IoT devices and fog nodes are heterogeneous and distributed. The architecture is divided into an IoT layer, a fog layer, and a Cloud layer. For simplicity of discussion, we consider the blockchain as part of the fog layer.

1) *IoT layer*: The IoT layer is composed of devices with varying resource capabilities and requirements from higher layers. We focus our approach on devices with limited computing/storage resources that require processing from the fog layer, and may send data to the Cloud layer for long-term storage. We define  $I$  as the set of IoT devices in the FISIE system. All IoT devices in  $I$  have at least enough resources to store the necessary encryption keys and to communicate with higher layers.

2) *Fog layer*: The fog layer is composed of fog nodes, oracles, and blockchain nodes for a smart contract-capable blockchain. We define  $F$  as the set of fog nodes and  $O$  as the set of oracles in the FISIE system. The fog nodes in  $F$  have varying resource capabilities, and some may have sufficient resources to run a light blockchain node [30], a blockchain oracle [41], or both. Blockchain nodes may exist separately, or within a fog node, i.e., blockchain-enabled fog nodes [19].

3) *Cloud layer*: The Cloud layer is composed of mega data-centers capable of long-term data storage and substantial computing power [46]. Data that require storage may come from the fog layer after it has been processed, or directly from the IoT layer.

## V. IIMSC - INITIALIZATION

Beginning in this section, and continuing in sections VI, VII and VIII, we define in detail the functionalities of IIMSC, including its initial parameters and tables. Every function of IIMSC takes a signature  $s$  as a final argument, which is validated via ECDSA before executing the function. Therefore, we omit the signature validation from the description of IIMSC functions. The functions and lookup tables of IIMSC are designed to 1) facilitate the mutual authentication process, 2) provide security and accountability to the IoT-fog service payment process, and 3) enable incentive and penalty mechanisms for fog integrity. A summary of all IIMSC functions are provided in Table II and are described in future sections.

Table II: A summary of IIMSC functions

<b>Initialisation</b>	Initialisation()
<b>Registration</b>	IoT_registration(Ether $E_u$ ) Fog_registration(Ether $E_u$ ) Oracle_registration()
<b>Funds</b>	IoT_add_funds(Ether $E_u$ ) IoT_withdraw_funds(float $u$ ) Fog_withdraw_funds(float $u$ )
<b>Removal</b>	IoT_remove() Fog_remove()
<b>Payment</b>	IoT_fog_payment(float $d$ )
<b>Audit result</b>	Fog_reward(Address $a_f$ , Ring sign. $\mathcal{R}_\omega$ ) Fog_penalize(Address $a_f$ , Ring sign. $\mathcal{R}_\omega$ )

#### A. Lookup tables

Secure mutual authentication schemes rely on a trusted third-party to validate the identity of each authenticating member [23], [25]. Therefore, we propose that IoT devices and fog nodes register with respective IoT and fog lookup tables on the blockchain. The registration process uses ECDSA signatures to initially confirm the identity of the registering party. Hence, all registration information on the lookup tables are publicly accessible and pre-verified. To register with the blockchain, we require a payment deposit from IoT devices, and a collateral deposit from fog nodes. These deposited amounts are reflected in the lookup tables. Since our default implementation uses Ethereum, all mentions of payments, funds and deposits will use the Ether cryptocurrency [20].

For each IoT device  $i \in I$ , the fields in the IoT lookup table  $T_I$  are defined as

- **IoTAddress:** The address of the IoT device  $i$ , which is also the truncated hash of the IoT public key  $H(P_i)||_{20}$ .
- **AvailFunds:** The available funds of IoT device  $i$ . These funds are used to pay for fog services.

For record  $\mathbf{t}_i \in T_I$  of IoT device  $i$ , we denote these entries as  $\mathbf{t}_i.A$ , and  $\mathbf{t}_i.AF$  respectively.

For each fog node  $f \in F$ , the fields in the fog lookup table  $T_F$  are defined as

- **FogAddress:** The address of the fog node  $f$ , which is also the truncated hash of the IoT public key  $H(P_f)||_{20}$ .
- **Deposit:** The collateral deposit given by fog node  $f$ . A portion of the deposit may be lost as a penalty for failing a service audit.
- **AvailFunds:** The available funds of fog node  $f$ . Available funds come from IoT service payments and may be withdrawn at the fog node's discretion.
- **Reputation:** The reputation score of fog node  $f$ . This reputation score is updated based on the results of a service audit. IoT devices may choose which fog nodes to work with based on their respective reputation scores.

For record  $\mathbf{t}_f \in T_F$  of fog node  $f$ , we denote these entries as  $\mathbf{t}_f.A$ ,  $\mathbf{t}_f.AF$ ,  $\mathbf{t}_f.D$ , and  $\mathbf{t}_f.R$  respectively.

By default, the lookup tables are implemented on-chain. Alternatively, the tables may be placed off-chain and managed

by a *reverse oracle*, i.e., an outbound oracle that executes on behalf of the blockchain [47]. In this case, we add an additional column to both tables labeled 'LastUpdateHeader'. For each table record, the 'LastUpdateHeader' field contains the blockchain header associated with the latest record update. By referencing this hash in the lookup tables, we also enforce immutability on the values of the off-chain lookup table.

In addition, we define an on-chain oracle lookup table  $T_O$  to register any oracle that wishes to participate in the service auditing process.  $T_O$  has a single field **OracleAddress**, denoted  $\mathbf{t}_o.A$  for record  $\mathbf{t}_o \in T_O$  of oracle  $o \in O$ .

#### B. Initialization

The Initialization function is the constructor of IIMSC. It creates the IoT, fog and oracle lookup tables  $\{\text{IIMSC}.T_I, \text{IIMSC}.T_F, \text{IIMSC}.T_O\}$ , and sets the following parameters:

- the minimum, initial and maximum reputation scores  $\{\text{IIMSC}.R_{\text{Min}}, \text{IIMSC}.R_{\text{Init}}, \text{IIMSC}.R_{\text{Max}}\}$ , where  $\text{IIMSC}.R_{\text{Min}} \leq \text{IIMSC}.R_{\text{Init}} \leq \text{IIMSC}.R_{\text{Max}}$
- the reputation penalty and reward  $\{\text{IIMSC}.r^-, \text{IIMSC}.r^+\}$ , where  $\text{IIMSC}.r^- > \text{IIMSC}.r^+$
- the fog collateral deposit amount  $\text{IIMSC}.D$  and penalty deposit deduction  $\text{IIMSC}.d^-$

It is important that the reward  $r^+$  is smaller than the penalty  $r^-$  to deter fog nodes from behaving outside of what is expected.

#### C. IIMSC pooled funds

During registration process, IoT devices, fog nodes and the oracles each submit deposits, either for payments or as collateral. These funds are 'moved' during the payment and penalty processes. All funds deposited into IIMSC are pooled within the smart contract, and the individual token holdings are detailed in the lookup table for each device. Hence, any payments that occur through IIMSC have no actual transfer of payments between devices. Rather, the lookup table values are updated, and token changes are realized upon withdrawal.

### VI. IIMSC – IDENTITY MANAGEMENT

The objective of the identity management functions of IIMSC is to facilitate mutual authentication between IoT and fog. Prior to sending a request, the IoT device must authenticate a fog node by verifying it is registered in  $T_F$  and has a sufficient reputation score. Likewise, the fog node must authenticate the IoT device to ensure it is registered in  $T_I$  and has sufficient funds to pay the fog node.

#### A. Registration

Once IIMSC has initialized, any entity that wishes to partake in the FISIE system must first register with the blockchain.

1) *IoT registration:* The `IoT_registration` function takes an initial Ether deposit  $E_u$  of amount  $u > 0$  from IoT device  $i \in I$ . After ensuring  $a_i = H(P_i)||_{20}$  is not already in  $T_I$ , the values  $\mathbf{t}_i.A \leftarrow a_i$  and  $\mathbf{t}_i.AF \leftarrow u$  are added to new record  $\mathbf{t}_i \in T_I$ .

2) *Fog registration*: The `Fog_registration` function takes a deposit  $E_d$  of amount  $d$  for fog node  $f \in F$ , with  $d \geq \text{IIMSC}.D$ . An amount  $\text{IIMSC}.D$  is used for the deposit, and the remainder  $v = d - \text{IIMSC}.D$  is set as the initial available funds. The reputation of  $f$  is set to the initial reputation  $r = \text{IIMSC}.R_{\text{Init}}$ . After ensuring  $a_f = H(P_f)||_{20}$  is not already in  $T_F$ , the values  $t_f.A \leftarrow a_f$ ,  $t_f.D \leftarrow \text{IIMSC}.D$ ,  $t_f.AF \leftarrow v$  and  $t_f.R \leftarrow r$  are added to the new record  $t_f \in T_F$ .

3) *Oracle registration*: An oracle  $o \in O$  with public key  $P_o$  must register with the blockchain in order to securely communicate its oracle results. Therefore, we define a function `Oracle_registration` to create a record  $t_o \in T_O$  with  $t_o.A \leftarrow H(P_o)||_{20}$ .

### B. Removal

If an IoT device  $i \in I$  no longer wishes to be a part of the network, it may call the `IoT_remove` function. All available funds under the record  $H(P_i)||_{20}$  are returned to IoT device  $i$ , and the record is removed from  $T_I$ .

A fog node may request to exit the system, or it may be removed forcefully by IIMSC. In both cases, the `Fog_remove` function is executed. If the `Fog_remove` function is initiated by the fog node, then the remaining deposit and available funds under the record  $H(P_f)||_{20}$  is returned to fog node  $f$ , and the record is removed from  $T_F$ . If at any point the fog deposit reaches zero or the reputation score falls below  $\text{IIMSC}.R_{\text{Min}}$ , the `Fog_remove` function is automatically triggered, the remaining available funds and deposit (if any) are returned and the fog node is removed from  $T_F$ .

### C. Mutual Authentication

We outline the mutual authentication process between an IoT device  $i \in I$  and a fog node  $f \in F$ .

- 1) IoT device  $i$  sends a signature  $s_i$  to fog node  $f$ .
- 2) Fog node  $f$  will recover  $P_i$  from  $s_i$ , and query  $H(P_i)||_{20}$  as the address of  $i$  from the IoT lookup table  $T_I$ .
- 3) If the IoT address is found in step 2), continue with step 4). If not, **mutual authentication fails**.
- 4) Fog node  $f$  sends a signature  $s_f$  to IoT device  $i$ .
- 5) IoT device  $i$  will simultaneously
  - (a) recover  $P_f$  from  $s_f$  and query  $H(P_f)||_{20}$  as the address of  $f$  from the IoT lookup table  $T_F$ .
  - (b) verify that the reputation score  $t_f.R$  meets the reputation threshold  $i.R$  set by  $i$ .
- 6) If the fog address is found and the reputation threshold is met in step 5), continue to step 7). If not, **mutual authentication fails**.
- 7) IoT device  $i$  and fog node  $f$  have successfully completed mutual authentication, and established a symmetric key  $P_f k_i = P_i k_f$  for secured communication. They may begin to collaborate.

The mutual authentication process is summarized in Fig. 2.

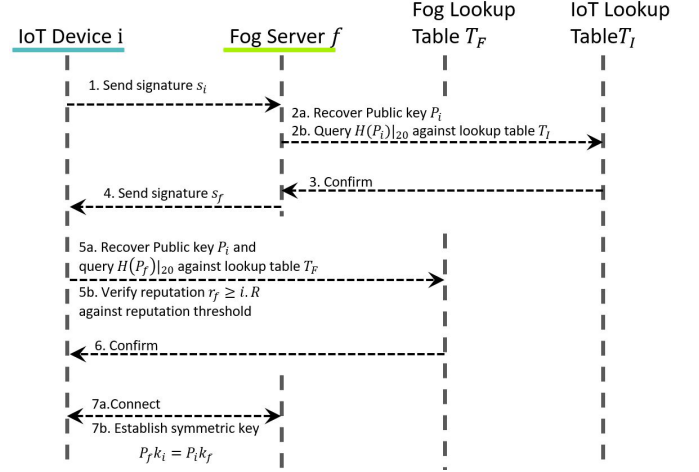


Figure 2: The mutual authentication process. Recall, every function takes a signature, from which the associated public key is recovered.

## VII. IIMSC – PAYMENT MANAGEMENT

The objective of the payment management functions of IIMSC is to streamline the IoT-fog service payment process, while also providing security and accountability. Since all payment transactions are posted on the blockchain, visibility of payment records can be used in the case of a payment dispute. Once an IoT device and fog node are mutually authenticated, the IoT device may send any computation requests to the fog node in exchange for a portion of deposited funds.

### A. Addition and withdrawal of funds

Once a device has registered with IIMSC, it may add or withdraw funds used for service payments.

1) *IoT funds*: After an IoT device  $i$  has registered with the blockchain, it may add additional funds to its available reserve. The `IoT_add_funds` function takes additional funds  $E_u$  of amount  $u > 0$ . The record  $t_i \in T_I$  where  $t_i.A = H(P_i)||_{20}$  is updated with  $t_i.AF \leftarrow t_i.AF + u$ . Similarly, the `IoT_withdraw_funds` may be used to withdraw an amount  $u \in (0, t_i.AF]$  from the available funds. Then, the record  $t_i \in T_I$  is updated with  $t_i.AF \leftarrow t_i.AF - u$  and  $E_u$  Ether is sent to IoT device  $i$ .

2) *Fog funds*: Once fog node  $f$  begins to service IoT requests, it will accumulate payments in its available funds. Fog node  $f$  may request to withdraw an amount  $u \leq t_f.AF$  through the `Fog_withdraw_funds` function. The `Fog_withdraw_funds` function takes an amount to withdraw  $u \in (0, t_f.AF]$ . The record  $t_f \in T_F$  where  $t_f.A = H(P_f)||_{20}$  is updated with  $t_f.AF \leftarrow t_f.AF - u$ , and  $E_u$  Ether is sent to  $f$ .

### B. IoT-Fog service and payment

Once the mutual authentication process is successfully completed and a symmetric key  $P_i k_f = P_f k_i$  has been established between  $i \in I$  and  $f \in F$ , IoT device  $i$  may request computational support from fog node  $f$  via symmetric



encryption. We outline the IoT-fog service process between IoT device  $i$  and fog node  $f$ .

- 1) IoT device  $i$  transmits a proposed payment  $d$ , a package  $g$ , and the signature  $s_i$  of the transaction to fog node  $f$ .
- 2) If  $f$  doesn't accept, it sends a 'reject' return statement OR lets the request time out. **The process ends.**
- 3) Else,  $f$  processes package  $g$  and gets result  $\tau$ .
- 4) The fog node  $f$  returns result  $\tau$  to IoT device  $i$ .
- 5) IoT device  $i$  triggers the `IoT_fog_payment` function with parameters: agreed payment  $d$  and IoT signature  $s_i$ .
- 6) The `IoT_fog_payment` function verifies signature  $s_i$ , and transfers an amount  $d$  from  $t_i.AF$  to  $t_f.AF$ ,  $t_i \in T_I$ ,  $t_f \in T_F$ .

The service and payment process is summarized in Fig. 3.

### C. Matching, Bargaining, and Disputes

The default implementations of mutual authentication and service payment described above are streamlined, without consideration of fog selection, price bargaining or payment disputes. In reality, there is room for flexibility to address these concerns in these processes.

Prior to mutual authentication, IoT devices must select to which fog node it wishes to contact for service. There exist many, more sophisticated matching algorithms for pairing IoT devices with fog nodes based on proximity and available fog resource capacity [48].

Once devices have been authenticated and the IoT device submits its request with proposed payment, the fog node may counter-offer. That is, the IoT device and fog node may enter into a round of bargaining to determine an agreed price [49]. Indeed, the matching and bargaining processes can even be combined into an auction-based process whereby fog nodes are matched by bidding on the IoT request [50].

Once the service process is complete and the fog node has returned the processed result  $\tau$  of package  $g$ , it is up to the IoT device to trigger the `IoT_fog_payment` function. If it does not within a reasonable amount of time, the fog node may start a dispute with a decentralized dispute resolution platform such as Kleros<sup>5</sup> [51]. Conversely, if the IoT device does not receive a response from the fog in a timely manner, the IoT device may start a dispute. These dispute resolution processes may prove to be costly for IoT devices and fog nodes, thus incentivizing timely processing of IoT requests, and timely triggering of the payment smart contract.

## VIII. IIMSC – INTEGRITY VERIFICATION

By occasionally checking the processing results of fog nodes, we can add a level of integrity and trust to the IoT-fog environment. We rely on trusted decentralized oracles to audit and verify the integrity of fog nodes. An oracle  $o \in O$  uses two key pairs – one registered as an 'IoT device' and one registered as an oracle. All audits are submitted by the address associated to the IoT lookup table  $T_I$ , so that fog nodes believe the audit is a normal IoT request. This is crucial to verify the natural behavior of a fog node when not under supervision.

<sup>5</sup><https://kleros.io/>

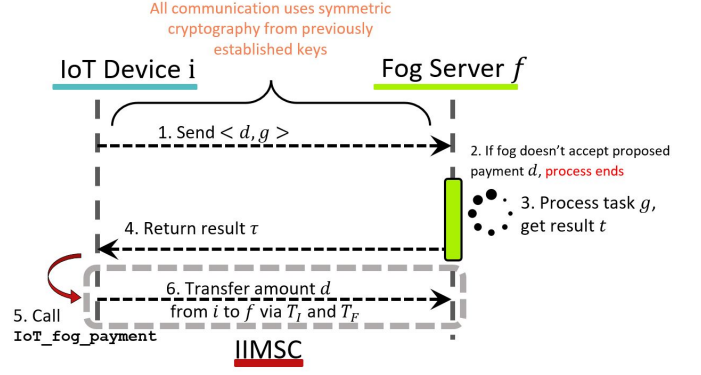


Figure 3: The IoT-fog processing and payment workflow.

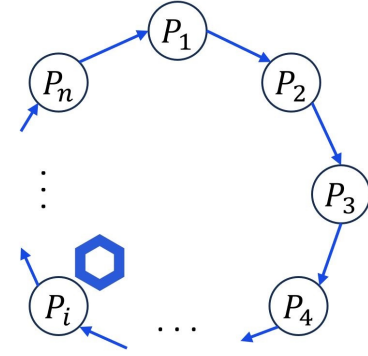


Figure 4: A ring signature, zero-knowledge proof of membership, hides the identity of the oracle among IoT.

### A. Ring Signature - Proof of Membership

Suppose oracle  $o \in O$  has two key pairs  $(P_\omega, k_\omega)$  and  $(P_\Omega, k_\Omega)$ . Prior to auditing, oracle  $o$  registers itself with the blockchain as an IoT device using key pair  $(P_\omega, k_\omega)$ , and as an oracle using key pair  $(P_\Omega, k_\Omega)$ . These key pairs result in different hashes across different tables. Therefore, the identity of oracle  $o$  on the IoT table is not known to fog nodes nor the blockchain. Hence, when submitting an audit result to the blockchain, the oracle must prove that the audit came from a valid IoT service request. In other words, it must prove that it belongs to the IoT lookup table without revealing the identity of its IoT address.

This problem relates to the class of zero knowledge proofs for set membership [35], which have been proven essential in blockchain applications [52]. Given the information publicly available in the IoT lookup table, we choose to implement a *ring signature scheme* [53].

Let  $P_I$  be the set of public keys from table  $T_I$ . We choose some enumerated subset  $\bar{P} \subseteq P_I$  containing  $P_\omega$  such that  $|\bar{P}| = n$  and  $\bar{P} = \{P_1, P_2, \dots, P_n\}$  where  $P_\omega = P_j$  for some  $1 \leq j \leq n$ .

An oracle with key-pair  $(P_\omega, k_\omega) = (P_j, k_j)$  builds a ring signature of message  $m$  over the elliptic curve of prime  $p$  and base point  $G$  as follows:

- 1) Choose a random integer  $q \in [0, p - 1]$ .
- 2) Calculate  $T_j = (x_j, y_j) = q \cdot G$ .
- 3)  $\forall i = 1, \dots, n, i \neq j$ , pick random integers  $\sigma_i \in [0, p - 1]$ .



- 4) for  $i = j + 1, \dots, n, 1, \dots, j - 1$ :
 
$$c_i = H(m|x_{i-1})$$

$$T_i = (x_i, y_i) = \sigma_i \cdot G + c_i \cdot P_i$$
- 5)  $c_j = H(m|x_{j-1})$
- 6)  $\sigma_j = q - c_j k_j$ .

Let  $\sigma = [\sigma_1, \dots, \sigma_n]$  and  $P = [P_1, \dots, P_n]$ . Oracle  $o$  submits  $(c_1, \sigma, P)$ .

1) *Verification*: The verifier (IIMSC) begins with  $c_1$ , and calculates

$$T_1 = (x_1, y_1) = \sigma_1 \cdot G + c_1 \cdot P_1$$

for  $i = 2, \dots, n$ :

$$c_i = H(m|x_{i-1})$$

$$T_i = (x_i, y_i) = \sigma_i \cdot G + c_i \cdot P_i$$

$$c'_1 = H(m|x_n).$$

The ring signature is accepted if  $c_1 = c'_1$ .

2) *Correctness*: By the original choice of  $T_j = q \cdot G$ , the final choice of  $\sigma_j$  'closes' the ring. That is,

$$T_j = \sigma_j \cdot G + c_j P_j$$

$$= (q - c_j k_j) \cdot G + c_j k_j \cdot G$$

$$= q \cdot G.$$

### B. Reward & Penalty Functions

IIMSC alters the deposit and reputation scores of fog nodes based on the results of a service audit. Both the `Fog_reward` and `Fog_penalize` functions take a ring signature  $\mathcal{R}_\omega$  from an oracle  $o \in O$ . If the ring signature is valid, then there exists an address in  $T_I$  that belongs to oracle  $o$ .

1) *Passed audit*: When a fog node  $f \in F$  passes a service audit sent by oracle  $o \in O$  with public key  $P_\Omega$ , the oracle calls the `Fog_reward` function which takes the fog address  $a_f$  and a ring signature  $\mathcal{R}_\omega$ . After verifying  $H(P_\Omega)||_{20}$  is in  $T_O$ ,  $a_f$  is in  $T_F$  and verifying the validity of ring signature  $\mathcal{R}_\omega$ , the function increases the fog reputation by an amount  $\text{IIMSC}.r^+$ , up to a maximum  $\text{IIMSC}.R_{\text{Max}}$ . That is,  $\mathbf{t}_f.R \leftarrow \min\{\mathbf{t}_f.R + \text{IIMSC}.r^+, \text{IIMSC}.R_{\text{Max}}\}$ .

2) *Failed audit*: When a fog node  $f \in F$  fails a service audit sent by oracle  $o \in O$  with public key  $P_\Omega$ , the oracle calls the `Fog_penalize` function which takes the fog address  $a_f$  and a ring signature  $\mathcal{R}_\omega$ . After verifying  $H(P_\Omega)||_{20}$  is in  $T_O$ ,  $a_f$  is in  $T_F$ , and verifying the validity of ring signature  $\mathcal{R}_\omega$ , the function 1) decreases the fog reputation by an amount  $\text{IIMSC}.r^-$ , and 2) decreases the deposit by an amount  $\text{IIMSC}.d^-$ , or to 0, whichever is higher. That is,  $\mathbf{t}_f.R \leftarrow \mathbf{t}_f.R - \text{IIMSC}.r^-$  and  $\mathbf{t}_f.D \leftarrow \max\{\mathbf{t}_f.D - \text{IIMSC}.d^-, 0\}$ . The lost deposit is distributed among the registered IoT devices. If the updated reputation  $\mathbf{t}.R$  falls below  $\text{IIMSC}.R_{\text{Min}}$ , or if the updated deposit  $\mathbf{t}.D$  reaches 0, then `Fog_remove` is automatically called on  $f$ .

### C. Service Audit

We define the IoT address  $a_\omega = H(P_\omega)||_{20}$  and oracle address  $a_\Omega = H(P_\Omega)||_{20}$  as the two addresses used by oracle  $o \in O$  for fog and blockchain communication respectively.

- 1) Oracle  $o$  and fog node  $f$  mutually authenticate and establish a symmetric key  $P_\omega k_f = P_f k_\omega$ .

- 2) Oracle  $o$  sends a package  $g$  to  $f$  following the process in section VII-B.
- 3) Simultaneously, oracle  $o$ 
  - a) waits for request response  $\tau_f$  from  $f$ .
  - b) calculates the expected output  $\tau_\omega$  of  $g$ .
- 4) Oracle  $o$  computes a ring signature  $\mathcal{R}_\omega = \{c_1, \sigma, P\}$  and compares the fog result  $\tau_f$  with the expected result  $\tau_\omega$ .
  - a) If  $\tau_f = \tau_\omega$ , fog node  $f$  has passed the service audit. Oracle  $o$  calls the `Fog_reward` function with fog address  $a_f$ , oracle signature  $s_\Omega$ , and ring signature  $\mathcal{R}_\omega$ .
  - b) Else, if  $\tau_f \neq \tau_\omega$ , and  $f$  has failed the service audit. Oracle  $o$  calls the `Fog_penalize` function with fog address  $a_f$ , oracle signature  $s_\Omega$ , and ring signature  $\mathcal{R}_\omega$ .

1) *Oracle payment*: When an oracle  $o$  executes a service audit, it takes time and uses processing resources for the benefit of the FISIE system. In addition, since oracle  $o$  is disguising a service audit as an IoT service request, it must pay a service fee to the audited fog node. In both cases, the oracle should be fairly compensated and reimbursed for its efforts.

By default, IIMSC takes a service fee from IoT devices whenever a call to `IoT_fog_payment(d)` is made. That is, IIMSC takes a small portion of the service payment  $d$  as the service fee. These fees are pooled by IIMSC. A portion of the pool is used to pay the oracles, and the rest is used to pay the owners of the smart contract.

2) *Scheduling policy*: The audit scheduling policy defines how often oracles can execute service audits. By default, one oracle completes one service audit every  $\eta$  requests, where  $\eta$  is defined by IIMSC. A large  $\eta$  ensures that more than enough fees have been collected to pay the oracle fairly, but may not result in frequent enough service audits. In contrast, a small  $\eta$  results in more, frequent service audits, but would require larger fees to be taken from IoT service payments to cover the oracle costs. Other more sophisticated scheduling policies [54], [55] can be considered for service auditing that take into account the overall health of the system. This is left for future work.

## IX. IIMSC – PENALTY & INCENTIVE MECHANISMS

The lookup tables, IIMSC parameters, and integrity verification functions are used to provide incentives and penalties for fog nodes to encourage integrity.

### A. Fog monetization

An IoT device  $i \in I$  may request service from a fog node, in exchange for a proposed payment  $d$ , where  $d \leq \mathbf{t}_i.AF$ . That is, the IoT device has sufficient available funds to satisfy the proposed payment. Once a fog node has serviced an IoT request, the IoT device pays the fog node for its services. This IoT payment provides a **monetary incentive** to fog nodes to service IoT. A service payment from an IoT device is deducted from its available funds in  $T_I$ , which is the total of all previously deposited and unspent funds, in IIMSC, from the IoT during or after registration.

### B. Fog collateral deposit

Upon registry, a fog node  $f \in F$  has collateral deposit  $\mathbf{t}_f.D = \text{IIMSC}.D$ . Periodically, a service audit is sent out to fog node  $f$  by an oracle posing as an IoT device. The fog node, unaware the request is from an oracle, would respond to the request normally, either with a correct or faulty response. If the response is incorrect, i.e., fog node  $f$  has failed the service audit, then a portion of the fog deposited funds  $\mathbf{t}_f.D$  are deducted from  $T_F$  and redistributed to IoT. This loss of collateral provides a **monetary penalty** to fog nodes if they fail a service audit. If a fog node loses its entire deposit, i.e.,  $\mathbf{t}_f.D = 0$ , then the fog node is removed from  $T_F$ , and hence, from the FISIE system.

By default, the collateral deposit amount  $\text{IIMSC}.D$  is fixed, and any additional deposit is converted to available funds. Alternatively, a possible implementation of IIMSC could allow for flexible deposit amounts, and a more sophisticated deposit deduction or reduction mechanisms [56]. For example, IIMSC could decrease the required deposit from long-term behaving fog nodes. In this case, the additional deposit over the newly reduced deposit threshold is converted to available funds that the fog node may withdraw. This implementation provides an additional incentive to fog nodes to behave properly over the long-term.

### C. Fog reputation

Fog nodes are given a reputation score in lookup table  $T_F$ . The reputation score is updated by IIMSC based on the results of a service audit. That is, the reputation in  $T_F$  is decreased if a fog node fails a service audit, and is increased if it passes. An IoT device may filter the fog nodes based on their reputation score before choosing where to send a request. Therefore, it is beneficial to the fog node to always behave properly, as to have a higher reputation and the possibility for more IoT requests, i.e., IoT payments. This provides a **service incentive** to fog nodes if they pass a service audit. Conversely, a loss of reputation provides a **service penalty** to fog nodes if they fail a service audit. Every fog node begins at the same initial reputation score  $\text{IIMSC}.R_{\text{Init}}$ , can increase up to a fixed maximum score  $\text{IIMSC}.R_{\text{Max}}$ , and is removed from the system if the score falls below a minimal reputation threshold  $\text{IIMSC}.R_{\text{Min}}$ .

By default, IIMSC will decrease and increase the reputation score by a fixed  $\text{IIMSC}.r^-$  and  $\text{IIMSC}.r^+$  respectively, where  $\text{IIMSC}.r^- > \text{IIMSC}.r^+$ . However, more sophisticated strategies for calculating [57] and updating [58] reputation score can be used, taking into account factors such as the number of IoT requests, and service level agreement (SLA) compliance [59].

### D. Fog deposit distribution

If a fog node fails an audit, an amount  $d^-$  is deducted from its collateral deposit on the fog lookup table  $T_F$ . The associated Ether  $E_{d^-}$  is still attached to IIMSC. Hence, we choose to distribute the amount  $\text{IIMSC}.d^-$  amount among a subset of IoT devices  $\bar{I} \subseteq I$  by updating available funds in the lookup table  $T_I$  by an equivalent amount. That is, for each  $i \in \bar{I}$ , we

select an amount  $d_i > 0$  such that  $\sum_{i \in \bar{I}} d_i = \text{IIMSC}.d^-$ , and increase the available funds  $\mathbf{t}_i.AF \leftarrow \mathbf{t}_i.AF + d_i, \forall i \in \bar{I}$ .

By default, IIMSC distributes the deducted deposit from  $f \in F$  equally among all IoT devices by an amount  $d^-/n$  where  $n = |\bar{I}|$ . Other distributions strategies are possible, such as distributing only to IoT devices that have previously been serviced by  $f$  in either an equal or weighted manner.

## X. SIMULATION OF INTEGRITY

To the best of our knowledge, no other contribution provides both an incentive and a penalty mechanism to enforce the integrity of an IoT-fog system. We first discuss the security of the FISIE system. To determine the effectiveness of our proposed system, we execute an auditing simulation over a set of malicious nodes. We also define several auditing scheduling policies to compare their effectiveness amongst each other.

### A. Security Analysis – Discussion

The FISIE system streamlines the IoT-fog mutual authentication, service and payment processes while maintaining secure and accountable IoT-fog communication.

1) *Encrypted Communication*: All direct communication between IoT and fog is encrypted by a 256-bit ECC protocol. A 256-bit ECC key size ensures a high level of security comparable to a 2072-bit RSA key size – the former encryption standard [24], [60]. Furthermore, the `secp256k1` elliptic curve proposed is already used by most blockchains [42]. Hence, there is no need for external third parties to define the system cryptographic parameters [25], which preserves the security of the FISIE system.

2) *Lookup Tables*: For a particular IoT device or fog node, its address on the lookup table is its blockchain address, which is generated from the hash of its public key. When device  $A$  communicates with device  $B$ ,  $B$  verifies the address of  $A$  on the lookup tables by extracting the public key of  $A$  from its signature, which is generated by its secret key. That is, some third entity  $C$  cannot impersonate  $A$  unless  $C$  possess the secret key of  $A$ . Hence, the security of the mutual authentication and identity verification, is equivalent to the security of the 256-bit ECC protocol used [24].

3) *Payment*: All payments are recorded on the blockchain via IIMSC. Hence, any disputes that may arise can be verified against the blockchain to ensure the accuracy of all claims. Furthermore, since all entities are registered with IIMSC, future implementations could freeze IoT assets until a dispute is resolved, or add a reputation score to IoT devices to track how often they default on service payments.

4) *Ring Signature*: An oracle uses an IoT address  $a_\omega$  to pose as an IoT device when interacting with a fog node. The oracle then submits its audit results to IIMSC using an oracle address  $a_\Omega$ . The oracle audit submission includes a ring signature, which is a zero-knowledge proof of membership to prove the audit was done with an IoT address, without revealing which. Since all blockchain records are public, fog nodes can see which IoT devices are in the ring, and therefore could potentially be the oracle. The probability that any IoT address in the ring belongs to the oracle is  $1/n$  for a ring of

$n$  IoT addresses, which diminishes as  $n$  increases. However, a larger  $n$  requires more logging space on the blockchain and more verification computation from IIMSC, which could be costly. Hence, a balance is required to increase  $n$  as much as is reasonable. Alternatively, oracle  $o$  may send a ring signature  $\mathcal{R}_\omega$  with a large  $n$  to a secondary computing oracle such as TrueBit<sup>6</sup> for transparent verification. The computing oracle then sends the result to IIMSC. This process would allow for larger ring signatures, thus increasing audit security, without inflating the cost of the smart contract.

### B. Auditing sampling policies

For a set of fog nodes  $F$  and a size  $C$ , we sample a distinct subset  $F_C$  from  $F$  of  $C$  nodes. Service audits are executed over members in  $F_C$  either sequentially or simultaneously. The purpose of clustering fog nodes even when executing sequentially is to limit the number of repeated audits to the same fog node in a short time period. That is, a larger size  $C$  increases the expected time between audits to the same fog node. Each sampling policy defines a specific way that clusters are sampled.

1) *Random sampling*: Clusters of size  $C$  are sampled randomly without repetition from the set  $F$ . This sampling method weighs all members equally, and makes no distinction between fog nodes who have previously passed their service audits, and those who haven't.

2) *Weighted sampling*: This sampling assigns a weight to each fog node, based on their previous audit history. The weights are initialized to be uniform. If a fog node fails a service audit, the weight is increased, denoting that this fog node should be audited more often. Similarly, if a fog node passes a service audit, we decrease the weight, indirectly giving audit priority to all other fog nodes. In practice, this method would require an oracle to keep a separate internal log of fog audit history.

3) *BIBD sampling*: This method is based on combinatorial design theory [61] where we build a series of finite balanced sets [62] of fog nodes  $F_B$ . We define the  $(F, B, B)$ -balanced incomplete block design (BIBD) as a series of blocks, i.e., subsets of  $F$ , that are of size  $B$  and have each fog node appear in  $B$  distinct blocks. The  $(F, B, B)$ -BIBD is computed at the beginning of the simulation, and is re-computed every time a fog node is ejected from the system.

### C. Auditing cost

We assign each fog node  $f \in F$  a random ‘malicious rate’  $m_f \in [0.4, 1]$ , a probability that the next request response will be faulty. By the definition of `Fog_penalize`, a fog node penalty will deduct a portion of the fog deposit, and remove the fog node from the system once that deposit reaches 0. For our simulation, we set each fog deposit to 3 and enforce a penalty of -1. That is, a fog node is removed from the system if it fails 3 service audits. For this simulation, we suppose the malicious rate of each fog node does not change in response to an audit result. The simulation is executed 1000 times per

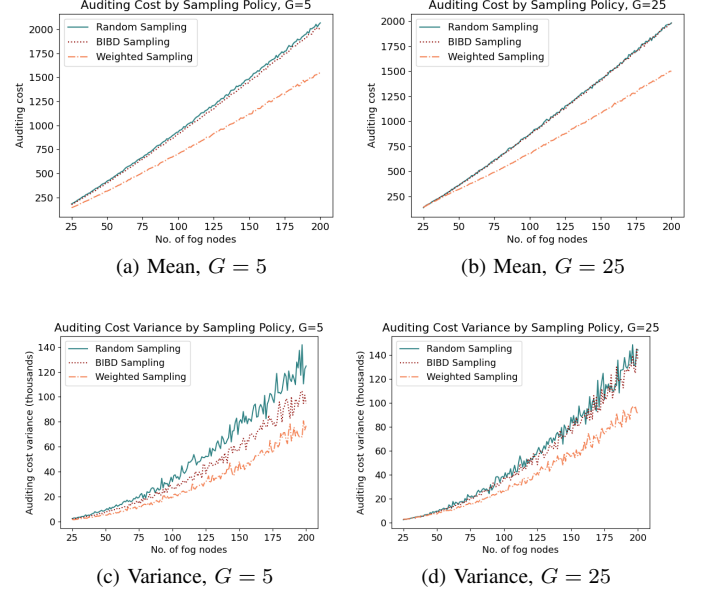


Figure 5: The means and variances of auditing costs necessary to expel all malicious nodes.

audit scheduling policy. We show the average of the results in Fig. 5a and 5b, and the variance of the results in Fig. 5c and 5d. From the results, the weighted sampling method consistently outperforms both random sampling and BIBD sampling in both mean and variance. Between random and BIBD methods, BIBD sampling performs marginally better. It is noted that using a larger cluster size  $C$  has no significant effect on the auditing cost.

### D. State of the system

Now we suppose that a fog node may alter its malicious rate in response to a service audit result. If a fog node fails a service audit, we decrease the malicious rate by a random fraction. Two scenarios may result: 1) the fog node adjusts its malicious rate slowly towards 0 and redeems its reputation score, thus staying in the system, or 2) the fog node fails more service audits before fully redeeming its reputation score, and is ejected from the system. In both cases, the integrity of the overall system increases. We observe this trade-off between the number of malicious nodes and the integrity of the system by simulating the state of the system over time. For these simulations, we set  $\text{IIMSC}.R_{\text{Min}} = 0$ , and  $\text{IIMSC}.R_{\text{Init}} = \text{IIMSC}.R_{\text{Max}} = 10$ .

1) *By malicious rate*: When a fog node fails a service audit, we decrease its malicious rate. Therefore, we expect the overall malicious rate to decrease over time. If a fog node is ejected from the system, then only fog nodes with lower malicious rates would stay in the system, further supporting our hypothesis. Indeed, as seen in Fig. 6a, there is a significant drop in the malicious rate over the first several audits. As expected, the overall reputation score initially drops, but slowly recovers once the majority of the fog nodes begin to behave properly.

<sup>6</sup><https://truebit.io/>

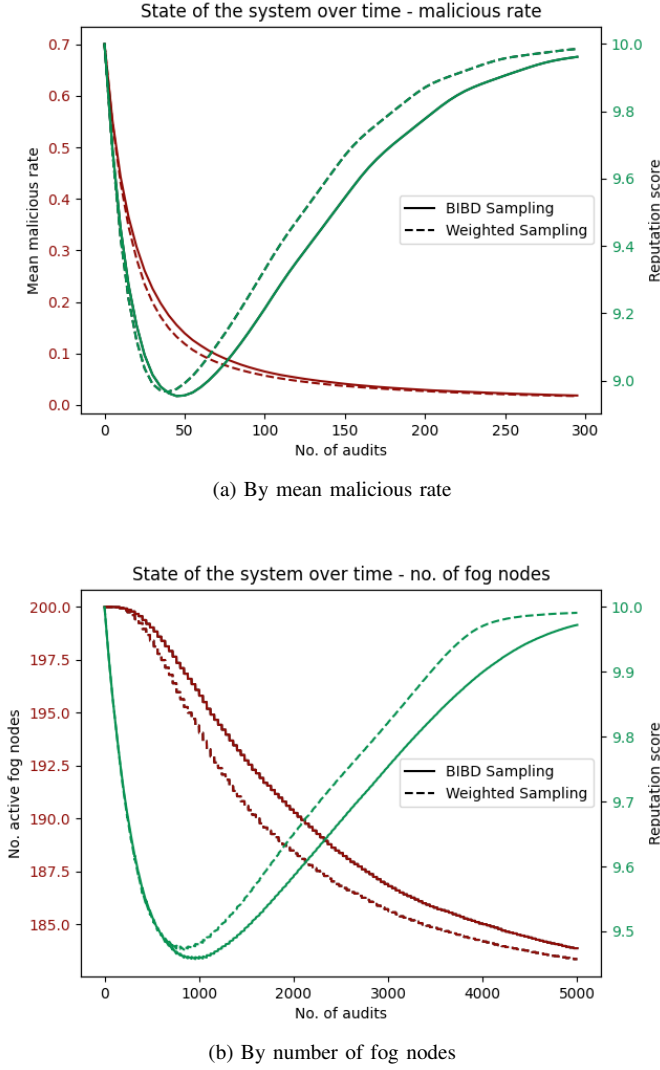


Figure 6: The integrity of the FISIE system over time

The redemption of the overall reputation score of the system is quicker with the weighted sampling method.

2) *By number of fog nodes:* Though not as drastic of a drop as the malicious rate, we do observe a decrease in the number of total fog nodes. Over time, fog nodes who have not sufficiently decreased their malicious rate are ejected from the system. A smaller pool of fog nodes, mostly composed of honorable fog nodes, will increase the total average reputation score. This is seen in the direct trade-off between increased reputation score and decreased number of fog nodes in Fig. 6b. The decrease in the number of active fog nodes is more drastic with the weighted sampling method. Over both simulation results, it is clear that the Weighted sampling method reaches full integrity in a shorter amount of time.

## XI. FUTURE WORK AND CONCLUSIONS

A key aspect of IoT security is ensuring the integrity of the fog nodes that interact with IoT. In this paper, we proposed a general architecture for heterogeneous IoT and blockchain-enabled fog nodes. We defined a smart contract-based system

for mutual authentication, monetization, and fog integrity enforcement. Finally, we analyzed the security of our system, and analysed the simulation results of our proposed service auditing over several audit scheduling policies. We found the weighted sampling method to increase system integrity in fewer total audits, hence lower blockchain cost.

In this study, the construction and analysis of the proposed system is theoretical. In future work, we will build a Proof of Concept model to study the feasibility of blockchain-enabled fog nodes, the actual incurred latency of mutual authentication and IoT task processing, and the actual behaviour of fog nodes over time. Furthermore, we will test various audit scheduling policies based on the real-time results of the system. Finally, we will include a data auditing mechanism in a public system to expand the scope of fog integrity. In a public IoT-fog environment with decentralized fog devices, integrity enforcement of fog will keep the environment safe and stable for IoT, and enable the expansion of IoT applications with the full cooperation of fog towards a real-world smart city [2].

## ACKNOWLEDGEMENT

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [CGS D - 558695 - 2021].

## REFERENCES

- [1] S. Kumar, P. Tiwari, and M. Zymbler, "Internet of things is a revolutionary approach for future technology enhancement: a review," *Journal of Big data*, vol. 6, no. 1, pp. 1–21, 2019.
- [2] C. Zhang, "Design and application of fog computing and internet of things service platform for smart city," *Future Generation Computer Systems*, vol. 112, pp. 630–640, 2020.
- [3] CISCO, "Fog computing and the internet of things: Extend the cloud to where the things are," tech. rep., 2015. Accessed: 2021-09.
- [4] G. L. Santos, P. T. Endo, M. F. F. da Silva Lisboa, L. G. F. da Silva, D. Sadok, J. Kelner, T. Lynn, *et al.*, "Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures," *Journal of Cloud Computing*, vol. 7, no. 1, p. 16, 2018.
- [5] C. Yu, B. Lin, P. Guo, W. Zhang, S. Li, and R. He, "Deployment and dimensioning of fog computing-based internet of vehicle infrastructure for autonomous driving," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 149–160, 2019.
- [6] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing," in *2015 International Conference on Information Networking (ICOIN)*, pp. 324–329, IEEE, 2015.
- [7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, 2012.
- [8] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, "Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study," *IEEE access*, vol. 5, pp. 9882–9910, 2017.
- [9] A. Giordano, G. Spezzano, and A. Vinci, "Smart agents and fog computing for smart city applications," in *International Conference on Smart Cities*, pp. 137–146, Springer, 2016.
- [10] I. Martinez, A. S. Hafid, and A. Jarray, "Design, resource management and evaluation of fog computing systems: A survey," *IEEE Internet of Things Journal*, 2020.
- [11] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–38, 2022.
- [12] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1210–1223, 2020.

- [13] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future generation computer systems*, vol. 82, pp. 395–411, 2018.
- [14] G. Cheng, Y. Chen, S. Deng, H. Gao, and J. Yin, "A blockchain-based mutual authentication scheme for collaborative edge computing," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 146–158, 2021.
- [15] L. Qiao, Y. Li, F. Wang, and B. Yang, "Lightweight integrity auditing of edge data for distributed edge computing scenarios," *Ad Hoc Networks*, vol. 133, p. 102906, 2022.
- [16] J. Wang, L. Wu, K.-K. R. Choo, and D. He, "Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1984–1992, 2019.
- [17] M. Debe, K. Salah, M. H. U. Rehman, and D. Svetinovic, "Monetization of services provided by public fog nodes using blockchain and smart contracts," *IEEE Access*, vol. 8, pp. 20118–20128, 2020.
- [18] H. Huang, X. Chen, Q. Wu, X. Huang, and J. Shen, "Bitcoin-based fair payments for outsourcing computations of fog devices," *Future Generation Computer Systems*, vol. 78, pp. 850–858, 2018.
- [19] G. Liu, J. Wu, and T. Wang, "Blockchain-enabled fog resource access and granting," *Intelligent and Converged Networks*, vol. 2, no. 2, pp. 108–114, 2021.
- [20] V. Buterin, "Ethereum white paper," tech. rep., 2013.
- [21] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A survey on access control in the age of internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4682–4696, 2020.
- [22] S. Algarni, F. Eassa, K. Almarhabi, A. Almalaise, E. Albassam, K. Alsubhi, and M. Yamin, "Blockchain-based secured access control in an iot system," *Applied Sciences*, vol. 11, no. 4, p. 1772, 2021.
- [23] R. Almadhoun, M. Kaddadha, M. Alhemeiri, M. Alshehhi, and K. Salah, "A user authentication scheme of iot devices using blockchain-enabled fog nodes," in *2018 IEEE/ACS 15th international conference on computer systems and applications (AICCSA)*, pp. 1–8, IEEE, 2018.
- [24] S. Kalra and S. K. Sood, "Secure authentication scheme for iot and cloud servers," *Pervasive and Mobile Computing*, vol. 24, pp. 210–223, 2015.
- [25] S. Singh and V. K. Chaurasiya, "Mutual authentication scheme of iot devices in fog computing environment," *Cluster Computing*, vol. 24, pp. 1643–1657, 2021.
- [26] A. A.-N. Patwary, A. Fu, S. K. Battula, R. K. Naha, S. Garg, and A. Mahanti, "Fogauthchain: A secure location-based authentication scheme in fog computing environments using blockchain," *Computer Communications*, vol. 162, pp. 212–224, 2020.
- [27] Y. Ding, Y. Li, W. Yang, and K. Zhang, "Edge data integrity verification scheme supporting data dynamics and batch auditing," *Journal of Systems Architecture*, vol. 128, p. 102560, 2022.
- [28] I. Zikratov, A. Kuzmin, V. Akimenko, V. Niculichev, and L. Yalansky, "Ensuring data integrity using blockchain technology," in *2017 20th Conference of Open Innovations Association (FRUCT)*, pp. 534–539, IEEE, 2017.
- [29] H. Tian, F. Nan, C.-C. Chang, Y. Huang, J. Lu, and Y. Du, "Privacy-preserving public auditing for secure data storage in fog-to-cloud computing," *Journal of Network and Computer Applications*, vol. 127, pp. 59–69, 2019.
- [30] A. Palai, M. Vora, and A. Shah, "Empowering light nodes in blockchains with block summarization," in *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)*, pp. 1–5, IEEE, 2018.
- [31] E. Reilly, M. Maloney, M. Siegel, and G. Falco, "An iot integrity-first communication protocol via an ethereum blockchain light client," in *2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT)*, pp. 53–56, IEEE, 2019.
- [32] A. Hafid, A. S. Hafid, and M. Samih, "Scaling blockchains: A comprehensive survey," *IEEE access*, vol. 8, pp. 125244–125262, 2020.
- [33] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, 2022.
- [34] V. Kapoor, V. S. Abraham, and R. Singh, "Elliptic curve cryptography," *Ubiquity*, vol. 2008, no. May, pp. 1–8, 2008.
- [35] E. Morais, T. Koens, C. Van Wijk, and A. Koren, "A survey on zero knowledge range proofs and applications," *SN Applied Sciences*, vol. 1, pp. 1–17, 2019.
- [36] D. Vujičić, D. Jagodić, and S. Randić, "Blockchain technology, bitcoin, and ethereum: A brief overview," in *2018 17th international symposium infoteh-jahorina (infoteh)*, pp. 1–6, IEEE, 2018.
- [37] O. Bouachir, M. Aloqaily, L. Tseng, and A. Boukerche, "Blockchain and fog computing for cyberphysical systems: The case of smart industry," *Computer*, vol. 53, no. 9, pp. 36–45, 2020.
- [38] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-peer Networking and Applications*, vol. 14, pp. 2901–2925, 2021.
- [39] A. Ouaddah, A. Abou El Kalam, and A. A. Ouahman, "Harnessing the power of blockchain technology to solve iot security & privacy issues," in *ICC*, pp. 7–1, 2017.
- [40] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, "Trustworthy blockchain oracles: review, comparison, and open research challenges," *IEEE access*, vol. 8, pp. 85675–85685, 2020.
- [41] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz, et al., "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," *Chainlink Labs*, vol. 1, pp. 1–136, 2021.
- [42] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, pp. 36–63, 2001.
- [43] M. Qu, "Recommended elliptic curve domain parameters," *Certicom Res., Mississauga, ON, Canada, Tech. Rep. SEC2-Ver-0.6*, 1999.
- [44] H. Mayer, "Ecdsa security in bitcoin and ethereum: a research survey,"
- [45] J. Thakur and N. Kumar, "Des, aes and blowfish: Symmetric key cryptography algorithms simulation based performance analysis," *International journal of emerging technology and advanced engineering*, vol. 1, no. 2, pp. 6–12, 2011.
- [46] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)*, pp. 105–110, IEEE, 2015.
- [47] R. Mühlberger, S. Bachhofner, E. Castelló Ferrer, C. Di Ciccio, I. Weber, M. Wöhrer, and U. Zdun, "Foundational oracle patterns: Connecting blockchain to the off-chain world," in *Business Process Management: Blockchain and Robotic Process Automation Forum: BPM 2020 Blockchain and RPA Forum, Seville, Spain, September 13–18, 2020, Proceedings 18*, pp. 35–51, Springer, 2020.
- [48] H. Tran-Dang and D.-S. Kim, "A survey on matching theory for distributed computation offloading in iot-fog-cloud systems: Perspectives and open issues," *IEEE Access*, 2022.
- [49] Y.-Y. Shih, C.-Y. Wang, and A.-C. Pang, "Fog computing service provision using bargaining solutions," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 1765–1780, 2019.
- [50] X. Peng, K. Ota, and M. Dong, "Multi-attribute based double auction towards resource allocation in vehicular fog computing," *IEEE Internet of Things Journal*, 2020.
- [51] J. Metzger, "Decentralized justice in the era of blockchain," *IJODR*, vol. 5, p. 69, 2018.
- [52] Z. Xu and L. Chen, "Div: resolving the dynamic issues of zero-knowledge set membership proof in the blockchain," in *Proceedings of the 2021 international conference on management of data*, pp. 2036–2048, 2021.
- [53] Y. F. Chung, Z. Y. Wu, and T. S. Chen, "Ring signature scheme for ecc-based anonymous signcryption," *Computer Standards & Interfaces*, vol. 31, no. 4, pp. 669–674, 2009.
- [54] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *2011 sixth annual ChinaGrid conference*, pp. 3–9, IEEE, 2011.
- [55] J. M. Schopf and F. Berman, "Stochastic scheduling," in *Proceedings of the 1999 ACM/IEEE Conference on Supercomputing*, pp. 48–es, 1999.
- [56] R. Kozhan and G. Viswanath-Natraj, "Decentralized stablecoins and collateral risk," *WBS Finance Group Research Paper*, 2021.
- [57] H. Yu, Z. Shen, C. Miao, C. Leung, and D. Niyato, "A survey of trust and reputation management systems in wireless communications," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1755–1772, 2010.
- [58] E. Bellini, Y. Iraqi, and E. Damiani, "Blockchain-based distributed trust and reputation management systems: A survey," *IEEE Access*, vol. 8, pp. 21127–21151, 2020.
- [59] R. Govindaraj, P. Govindaraj, S. Chowdhury, D. Kim, D.-T. Tran, and A. N. Le, "A review on various applications of reputation based trust management," *International Journal of Interactive Mobile Technologies*, vol. 15, no. 10, 2021.
- [60] A. Hamza and B. Kumar, "A review paper on des, aes, rsa encryption standards," in *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, pp. 333–338, IEEE, 2020.
- [61] D. R. Stinson, *Combinatorial designs: constructions and analysis*, vol. 480. Springer, 2004.

- [62] R. Bose and S. Shrikhande, “On the composition of balanced incomplete block designs,” *Canadian Journal of Mathematics*, vol. 12, pp. 177–188, 1960.



**ISMAEL MARTINEZ** received the B.Sc. degree in mathematics and computer science from Simon Fraser University, BC, Canada in 2016. He is currently a fourth-year Ph.D candidate at the University of Montreal, QC, Canada in Computer Science and Operations Research. His research interests are in network design of fog infrastructures, and security of IoT-fog infrastructures via zero-knowledge proofs and blockchain.



**ABDELHAKIM SENHAJI HAFID** is a Full Professor at the University of Montreal. He is the founding director of Network Research Lab and Montreal Blockchain Lab. Prof. Hafid published over 260 journal and conference papers; he also holds three US patents. Prior to joining U. of Montreal, he spent several years, as senior research scientist, at Bell Communications Research (Bellcore), NJ, US working in the context of major research projects on the management of next generation networks. Prof. Hafid has extensive academic and industrial research experience in the area of the communication networks and distributed systems. His current research interests include Blockchain scalability and security, Blockchain disruption of various industry segments, IoT, Fog/edge computing, and intelligent transport systems.



**MICHEL GENDREAU** is the Department Chair and Professor of operations research in the Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montreal, QC, Canada. He has published more than 300 papers in peer-reviewed journals and conference proceedings. He is also the Co-Editor of six books. His main research interest focuses on the application of operations research methods to energy planning and to the management of transportation and logistics systems. He was the chair holder of the NSERC/Hydro-Québec Industrial Chair on the Stochastic Optimization of Electricity Generation from 2009 to 2015. In 2001, he received the Merit Award of the Canadian Operational Research Society in recognition of his contributions to the development of O.R. in Canada. He was elected Fellow of INFORMS in 2010. In 2015, he received the prestigious Robert Herman Lifetime Achievement Award of the Transportation Science & Logistics Society of INFORMS.