Speeding at the Edge: An Efficient and Secure Redactable Blockchain for IoT-Based Smart Grid Systems

Youshui Lu, *Member, IEEE*, Xiaojun Tang, *Member, IEEE*, Lei Liu, *Member, IEEE*, F. Richard Yu, *Fellow, IEEE*, and Schahram Dustdar, *Fellow, IEEE*

***Abstract*—As a promising approach to extending cloud resources and services, blockchain-enabled Internet of Things (IoT)-based smart grid edge computing has attracted much attention. However, the edge node's resource-constraint nature makes it difficult to store the entire chain as the sensing IoT data volume increases. To address this issue, we propose an FS scheme, a fast and secure multithreshold trapdoor Chameleon hash scheme which serves as the basis for block substitution at the edge nodes to solve the storage limitation problem. The FS scheme is used to achieve a consensus-based block substitution, which allows t-out-of-n edge nodes to compute a hash collision collaboratively to reliably substitute a historical block without leaking the randomness R. Also, inspired by the rationale of fast polynomial interpolation, we optimize the FS scheme to FS-I to reduce the time complexity from $\mathcal{O}(nt)$ to $\mathcal{O}(t\log^2 t)$. In addition, we further optimize FS-I to FS-II by using a fast Fourier transform (FFT) to dramatically improve the computational efficiency of Lagrange interpolation, which leads to a significant improvement in terms of block substitution performance. Finally, We provide security analysis and evaluate the performance through comprehensive experiments and the results show that FS can achieve up to several magnitudes better than DTTCH. The results also demonstrate that the FS scheme can provide high service quality for large-scale IoT-based smart grid systems.**

***Index Terms*—Blockchain, Chameleon hash (CH), edge computing, industrial Internet of Things (IoT), redactable blockchain.**

I. Introduction

With the development of computing technologies, Internet of Things (IoT)-based edge computing have been applied to the smart grid systems to improve the reliability, quality, and flexibility of the energy transmission while reducing the communication latency between different nodes. IoT-based edge computing [2], [3] aims at providing computational resources close to billions of end devices at the edge of the smart grid network, it scales to a massive number of sites and is a cost efficient way to achieve scalability than cloud services. However, its resource-constraint and heterogeneous nature do also bring security challenges. During the data transmitting process, attacks, such as sniffer attacks and jamming attacks could disable the connections by congesting the network. Further, the data in the edge networks are divided into many parts and stored in different storage domains, which may cause data reliability issues [1], [4].

Meanwhile, many scholars use blockchain as a building block to integrate edge computing in IoT-based smart grid systems [5], [6], [7]. With the blockchain technology, it is possible to build a distributed control at dozens of edge nodes. Thanks to the chain structure and consensus process, blockchain can transparently protect the collected IoT data's accuracy, consistency, and validity. The integration of edge computing and blockchain seems to be a win–win solution that can provide secure and reliable services.

However, integrating blockchain and edge computing still suffer data storage capacity problems. Although the edge node could provide relatively large storage, as the collected IoT data increases, the storage required for blockchain ledger is ever-growing. As a result, the edge nodes will eventually consume the entire storage. The size of the current Bitcoin chain is more than 380 GB in size as of the end of February 2022 [8], while in the IoT-based smart grid settings, the size of the chain could be even larger. To address this problem, approaches, such as Ethereum differentiate full and light nodes. Only the full node stores the entire chain, while the light node only stores the block's headers.

Manuscript received 11 August 2022; revised 20 December 2022 and 17 February 2023; accepted 4 March 2023. Date of publication 7 March 2023; date of current version 7 July 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFF0903400; in part by the Open Research Fund Program of Key Laboratory of Agricultural Blockchain Application, Ministry of Agriculture and Rural Affairs under Grant 2022KLABA01; in part by the China Postdoctoral Science Foundation under Grant 2022M712535; in part by the Natural Science Foundation Research Program of Shaanxi Province under Grant 2023-JC-QN-0749; in part by the Key Research and Development Program of Shaanxi Province under Grant 2021GXLH-Z-054; in part by the Key Research and Development, Guangdong High Level Innovation Research Institution Project under Grant 2021B0909050008; and in part by the Transformation Program of Qinghai Province under Grant 2021-GX-112. *(Corresponding author: Xiaojun Tang.)*

Youshui Lu is with the School of Electrical Engineering, Xi'an Jiaotong University, Xi'an 710049, China, and also with the Key Laboratory of Agricultural Blockchain Application, Ministry of Agriculture and Rural Affairs, Agricultural Information Institute, Chinese Academy of Agricultural Sciences, Beijing 100081, China (e-mail: yolu6176@uni.sydney.edu.au).

Xiaojun Tang is with the School of Electrical Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: xiaojun_tang@xjtu.edu.cn).

Lei Liu is with the Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China (e-mail: tianjiaoliulei@163.com).

F. Richard Yu is with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada (e-mail: richard.yu@carleton.ca).

Schahram Dustdar is with the Distributed Systems Group, Vienna University of Technology, 1040 Vienna, Austria (e-mail: dustdar@dsg.tuwien.ac.at).

Digital Object Identifier 10.1109/JIOT.2023.3253601

Nevertheless, such a fashion brings centralization risk since the full nodes may be malicious and the light nodes have no efficient detection methods [9], [10]. Also, the edge node should be able to verify transactions and blocks, therefore, it is required to store the entire chain. Another solution is to overwrite the original on-chain data directly, but doing so would break the blockchain's tamper-resistance property. The existing approaches cannot effectively solve storage issues.

To address the storage limitation at the edge, our high-level idea to use redactable blockchain rationale [13], [27], [28], [29] to construct a hybrid node for the edge computing servers. So that in the collaborative edges, it enables the block substitutions of the historical block seamlessly when the storage of the edge server reaches the maximum, thus, mitigating the storage constraint of the edge.

Meanwhile, Chameleon hash (CH)-based redactable approaches [11], [12], [13], [16], [17], [18], [19], [20] are becoming popular in both academia and industrial fields. CH scheme is a one-way hash function that contains a public key and a trapdoor, where hashing is parametrized by public key pk. As long as the trapdoor sk is not revealed, it is difficult to find a hash collision. In the contrast, if sk is revealed to the adversary, the arbitrary collision can be found efficiently. In this article, we use the dynamic threshold trapdoor CH scheme (DTTCH) [20] as a building block to enable the block substitution process. Although DTTCH can tolerate a considerable number of malicious nodes by replacing a single sk with multiple secret keys, direct using DTTCH still faces two main challenges.

1) *Vulnerability Toward Randomness R:* With a publicly accessible R, it is getting easier for the adversaries to find the arbitrary collision.
2) *High Computational Overheads:* Compared with traditional block creation, overwriting the historical block by using the DTTCH scheme incurs high computational overheads, which is not compatible with a large-scale IoT-based smart grid system.

To address the aforementioned challenges, we propose the FS scheme, a fast and secure multithreshold trapdoor CH scheme which serves as the basis for block substitution in the edge nodes under IoT-based smart grid systems. The FS scheme is used to achieve a consensus-based block substitution, which allows a t-out-of-n edge node to compute a hash collision collaboratively to reliably substitute a historical block without leaking the randomness R. Also, inspired by the rationale of fast polynomial interpolation [14], we optimize the FS scheme to FS-I to reduce the time complexity from $\mathcal{O}(nt)$ to $\mathcal{O}(t\log^2 t)$. In addition, we further optimize FS-I to FS-II by using fast Fourier transform (FFT) [15] to dramatically improve the computational efficiency of the Lagrange interpolation, which leads to a significant improvement in terms of block substitution performance.

Finally, we conduct a security analysis and a comprehensive performance evaluation throughout the proof-of-concept implementation which we realized in Python and C++. The evaluation results show that FS-II's performance can be improved by several orders of magnitude compared with the DTTCH approach, it demonstrates that our proposed method is viable for a large-scale IoT-based smart grid system.

The main contributions of this work are summarized as follows.

1) We propose a redactable blockchain for the IoT-based smart grid systems, which can substitute the historical data and transaction of the earlier blocks without affecting the integrity of the original chain, addressing the storage barrier for the edge nodes in large-scale IoT-based smart grid systems.
2) We propose an FS scheme to achieve consensus-based block substitutions, which allows a t-out-of-n edge node to compute a hash collision collaboratively to reliably substitute a historical block without leaking the randomness R.
3) We optimize the FS scheme to FS-I by borrowing the rationale of fast polynomial interpolation to reduce the time complexity from $\mathcal{O}(nt)$ to $\mathcal{O}(t\log^2 t)$. To further optimize FS-I to FS-II, we use FFT to dramatically improve the computational efficiency of the Lagrange interpolation, leading to a significant improvement in terms of block substitution performance.
4) We instantiate a prototype implementation of the FS scheme, and evaluate the performance of algorithms and block substitution operations through comprehensive experiments. The results demonstrate that our proposed method is practical when applied to a large-scale IoT-based smart grid system.

II. Related Work

The CH algorithm was first proposed by Krawczyk and Rabin [11]. It is hashed by a public key, and the trap door is equivalent to a hash backdoor and is hidden. Users can use trapdoors to compute collisions of different messages with the same hash. In an editable blockchain, the user can replace the data in the block with any content without breaking the chain hash continuity by using the CH's private key. In the editable blockchain method proposed by the author, the trapdoor of CH can be calculated by multiparty calculation under the decentralization setting, that is, the process of block modification can be completed by multiparty security calculation. To ensure the traceability of the modification operation, the method sets a flag in the modified block to indicate the data modification operation and to ensure the transparency of the modification operation.

Ateniese et al. [13] first proposed a redactable blockchain by using CH so that data in the block can be replaced without resulting hard forks. Also, Derler et al. [16] proposed a fine-grained transaction-level redactable blockchain, their scheme uses a policy-based CH with ephemeral trapdoors. It allows any entity that satisfies the defined policy can then find collisions for a given hash. However, it is vulnerable for the entity to rewrite transactions being compromised or the single ephemeral trapdoor and long-term trapdoor are leaked, and data on the chain can be modified arbitrarily. Moreover, Deuber et al. [17] proposed the first redactable blockchain in the permissionless setting, once an edit operation is proposed

TABLE I
COMPARISON WITH THE RELATED WORK

	Type	Authorization	Security	Efficiency	Compatibility with IoT applications
CH [13]	Permissionless	×	×	★	×
PCH [17]	Permissionless	√	×	★	×
TTCH [18]	Permissioned	√	×	★	×
TCH [19]	Permissioned	√	√	★★	√
DTTCH [20]	Permissioned	√	×	★★	×
DTCH [22]	Permissioned	√	√	★★	×
Our Work	Permissioned	√	√	★★★	√

by a user, the protocol starts a consensus-based voting process, and only after obtaining enough votes for approving the redaction, the new block is updated. Also, the old state of the block is maintained for validation purposes. However, the voting period is very long and will cause high latency (it takes almost a few days to reach a consensus and publish a redaction block). In addition, Zhang et al. [18] proposed threshold trapdoor CH (TTCH) by using multiple secret keys instead of a single fixed secret key to finding collisions, it could improve the security of the trapdoor, but the computational costs are large.

As shown in Table I, another work proposed by Huang et al. [19] is threshold CH (TCH), especially, for Industrial IoT (IIoT) settings, which allows a block rewrite by a group of authorized entities. However, in the rewrite process of TCH, the system cannot tolerate malicious sensors, because the entire system will fail if one node is compromised. Similarly, Lu et al. [20] proposed an Acce-chain which enables the rewrite operation governed by a group of edge nodes, thus, making the rewrite process more controllable. Also, the authors present a DTTCH scheme which can easily find a collision through a group of trapdoor keys rather than relying on a single trapdoor key, it has better resilience to key compromise. In addition, the rewrite process is consensus-based which can tolerate a considerable number of faulty nodes. Moreover, even if an entity is executing a rewrite, the rest nodes in the system remain unlinkable, so it will not increase the latency. However, the newly generated randomness during rewrite operation will be exposed to the public which brings vulnerabilities to the system, also it is not practical for a large-scale IoT-based smart grid setting.

At the same time, Ateniess and others cooperated with Accenture [21] to implement a prototype system of an editable blockchain based on the Hyperledger Federation chain. The author uses a double-trapdoor CH [22] with an active trapdoor to encrypt the active trapdoor through an attribute-based encryption scheme, which enables a fine-grained modification permission assignment. Users generate an active trapdoor for the hash each time they hash with the CH public key. Other users who hold a CH long-term trapdoor can only compute collisions for a hash if they also get an active trap for that hash. Compared with the single trap in the traditional scheme, the CH with an active trap improves the flexibility of the trap and the security of computing collisions by setting a long-term trapdoor and a short-term trapdoor. This means that the modifying party needs the authorization of the encrypting party to calculate collisions and solves the problem that any block can be modified only by a long-term trap in blockchain modification.

III. PRELIMINARIES

This part presents the preliminaries on the fundamental cryptographic constructs for our algorithm design.

Notations: Let g denote a generator of a cyclic group $\mathbb{G}$ of order p. For a λ-bit prime number p, an algorithm is efficient if it runs in probabilistic polynomial time (PPT) in the length of its input. We denote the integer modulo p as $\mathbb{Z}_p$ and $r \xleftarrow{R} \mathbb{Z}_p$ denote that r is chosen uniformly at random from $\mathbb{Z}_p$, ab represent the multiplication of two integers $a \in \mathbb{Z}_p$ and $b \in \mathbb{Z}_p$.

A. Computational Assumptions

1) Discrete Logarithm Assumption: Given the cyclic group generation algorithm $(\mathbb{G}, p, g) \leftarrow \mathbf{Gen}(\mathbf{1}^\lambda)$, we get the cyclic group $\mathbb{G}$, order p and the generator g. The discrete logarithm (DL) problem is by given $g, g^x \in \mathbb{G}$, $x \leftarrow \mathbb{Z}_p$, compute x.

Definition 1 (DL Assumption): We say the DL problem is hard relative to $\mathbb{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$ there exists a negligible function *negl* such that

$$\mathrm{Adv}^{\mathrm{DL}}_{\mathbb{G},\mathcal{A}}(\lambda) = \Pr\big[(\mathbb{G}, p, g) \leftarrow \mathbf{Gen}\big(\mathbf{1}^\lambda\big) \; a \leftarrow \mathbb{Z}_p : \mathcal{A}\big(g, p, g^x\big) = x\big] = negl(\lambda).$$

2) Computational Diffe–Hellman Assumption: Given the cyclic group generation algorithm $(\mathbb{G}, p, g) \leftarrow \mathbf{Gen}(\mathbf{1}^\lambda)$, we get the cyclic group $\mathbb{G}$, order p and the generator g. Computational Diffe–Hellman (CDH) problem is by given (g, g^a, g^b), $a, b \leftarrow \mathbb{Z}_p$, compute g^{ab}.

Definition 2 (CDH Assumption): We say the CDH problem is hard relative to $\mathbb{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$ there exists a negligible function *negl* such that

$$\mathrm{Adv}^{\mathrm{CDH}}_{\mathbb{G},\mathcal{A}}(\lambda) = \Pr\big[(\mathbb{G}, p, g) \leftarrow \mathbf{Gen}\big(\mathbf{1}^\lambda\big) \; a, b \leftarrow \mathbb{Z}_p : \mathcal{A}\Big(g, g^a, g^b\Big) = g^{ab}\Big] = negl(\lambda).$$

3) Divisible Computational Diffe–Hellman Assumption: Given the cyclic group generation algorithm $(\mathbb{G}, p, g) \leftarrow \mathbf{Gen}(\mathbf{1}^\lambda)$, we get the cyclic group $\mathbb{G}$, order p, and the generator g. Divisible computational Diffe–Hellman (DCDH) [26] problem is by given (g, g^a, g^b), $a, b \leftarrow \mathbb{Z}_p$, compute $g^{a/b}$.

Definition 3 (DCDH Assumption): We say the DCDH problem is hard relative to $\mathbb{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$ there exists a negligible function *negl* such that

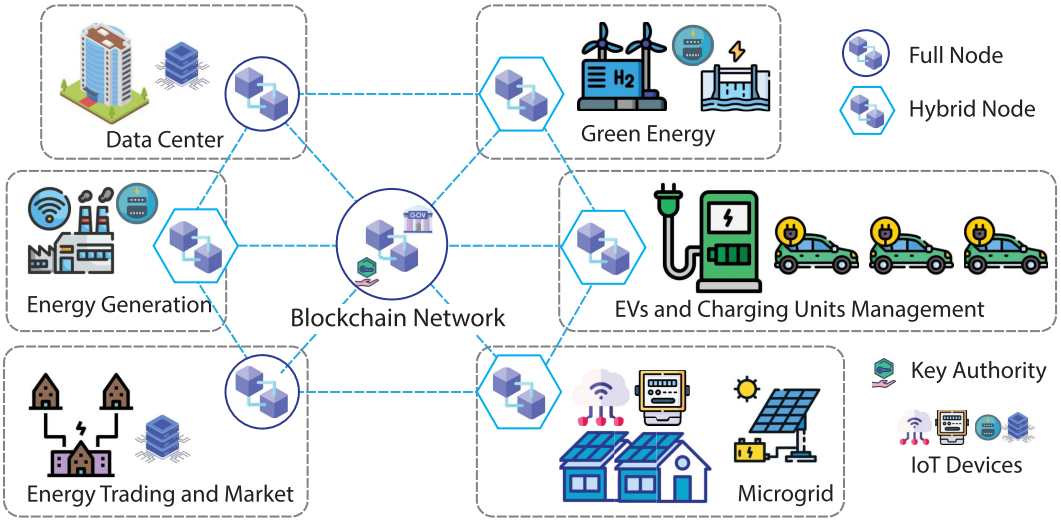

Fig. 1. Architecture of blockchain for IoT-based smart grid systems.

$$\mathrm{Adv}^{\mathrm{DCDH}}_{\mathbb{G},\mathcal{A}}(\lambda) = \Pr\Big[(\mathbb{G}, g, p) \leftarrow \mathbf{Gen}\big(\mathbf{1}^{\lambda}\big) \; a, b \leftarrow \mathbb{Z}_p : \mathcal{A}\Big(g, g^a, g^b\Big) = g^{a/b}\Big] = negl(\lambda).$$

B. Dynamic Threshold Trapdoor Chameleon Hash Scheme

In our previous work, we proposed the DTTCH scheme [20], which is using multiple secret keys instead of a single fixed secret key to find collisions. DTTCH is built on the bilinear group $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with order p. The functions satisfy the DCDH assumption.

1) *TT.Setup* $(1^{\lambda}) \rightarrow$ *(params):* The algorithm takes as input a security parameter 1^{λ} and outputs the system parameter $params = (\mathbb{G}_1, \mathbb{G}_2, p, g_1, g_2, H')$, g_1, g_2 is the generator of $\mathbb{G}_1$, $\mathbb{G}_2$, respectively.
2) *TT.Gen (params, t, n)* $\rightarrow$ *(hk,* $(sk_1, \ldots, sk_n)$*):* The algorithm takes as input the system parameter *params*, a predefined threshold t and the total number of the entities in the system n, and outputs n secret keys $(sk_1, \ldots, sk_n)$ and the public hash key $hk = (h_1, h_2, \widehat{h})$.
3) *TT.Hash(m, hk)* $\rightarrow$ *(*$\hbar, R$*):* The algorithm takes as input a message m and a hash key hk, and outputs a hash $\hbar = h_1^r \widehat{h}^m$ and a randomness $R = g_1^r$.
4) *TT.HVer* $(m, \hbar, hk, R)$ $\rightarrow$ *(True or False):* The algorithm takes as input a committed message m, a hash $\hbar$, a hash key hk and a randomness R, and verifies through bilinear pairing function, and outputs $[True/False]$.
5) *TT.Sign* $(sk_i, m', \hbar) \rightarrow (\sigma_i)$*:* The algorithm takes as input a secret key sk_i, a message m' and the hash $\hbar$, and outputs the message m''s credential σ_i.
6) *TT.Hcol* $((\sigma_1, \ldots, \sigma_t), R, hk, \hbar, m) \rightarrow (\perp$ *or* $R')$*:* The algorithm takes as input t credentials $(\sigma_1, \ldots, \sigma_t)$, a randomness R, a public hash key hk, a hash $\hbar$ and a message m, the algorithm first verifies by running $\mathbf{TT.HVer}(m, \hbar, hk, R)$, if verified, the algorithm returns $\perp$. Otherwise, it parses each σ_i for $i \in [1, \ldots, t]$ and outputs a new randomness R'.

IV. System Overview

In this section, we first present a typical user case with blockchain-enabled IoT-based smart grid systems, where edge nodes have limited storage. Then, we discuss the threat model and identify the design goals.

A. Application Scenario

In a multilayer IoT-based smart grid system, there is a large amount of IoT data which is latency sensitive, the typical applications include demand response, energy trading, and so on. Traditional cloud outsourcing services can not well support those applications due to the large communication overheads. Applications using edge computing must allow quick state synchronization between different edge nodes to guarantee that the quality of services is not affected by switching of edge domains. For example, in demand response energy trading applications, where it is highly anticipated that the data (such as real-time electricity usage) used for pricing must be reliable and consistent so as to eliminate possible unexpected accidents. Integrating the blockchain technology can help protect data security and accountability in the smart grid network.

B. Architecture

The typical multilayer blockchain-based IoT-based smart grid system is illustrated in Fig. 1. It mainly consists of four entities.

1) *Full node (cloud server)* has massive storage and computational power. Full nodes store the entire blockchain.
2) *Hybrid node (edge)* has relatively limited computation and storage capacity. It is the intermediate storage layer between IoT-based smart grid devices and the Cloud Server. Hybrid Node in the proposed architecture only stores the blocks that contain the hot data (most frequently accessed) and the block headers of the entire chain.

3) *Key authority* is a trusted authority that responds by issuing secret keys to each entity in the system and publishing the genesis block.
4) *IoT devices* (e.g., actuators and sensors) are used for generating and collecting sensing data. As its highly resource-constrained nature, it only generates and uploads raw sensing data to the edge devices.

C. Main Operations

The main operations consist of block creation, block substitution, and data query. We provide a brief introduction in the following.

1) Block Creation: This operation proposes a block which stores the smart grid IoT sensing data and finally appends such block to the ledger after consensus.

2) Block Substitution: This operation substitutes the block with cold data by the generated block with hot data. It only takes place when the edge nodes reach their maximum storage limit.

3) Data Query: This operation allows the edge node to query from the full nodes. In this article, we mainly focus on the reliable block substitution part.

D. Security Model

In this article, we focus on the security of the process of substituting the historical blocks, we assume that the system can tolerate a maximum number of *n-t* Byzantine nodes. Also, we assume that the communication channel is secure and the adversary can not break the standard cryptographic assumptions and primitives, e.g., finding hash collisions or forging credentials. For our proposed **FS** scheme, we will consider the following three security properties.

1) *Indistinguishability:* Given a multithreshold CH $\Pi_{\mathbf{FS}}$ scheme, a randomness generated by **Hash**$(\cdot)$ is indistinguishable from a randomness generated by **Hcol**$(\cdot)$ from the set of all functions having the same domain $\mathcal{M}$.
2) *Collision-Resistance:* An adversary $\mathcal{A}$ is given oracle access to **Hcol**$(\cdot)$, the probability for $\mathcal{A}$ to forge a valid hash collision is negligible.
3) *Robustness:* An adversary $\mathcal{A}$ can control a maximum number of $t-1$ entities (t is the predefined threshold) and also is given oracle access to **Sign**$(\cdot)$ and **Hcol**$(\cdot)$, the probability for $\mathcal{A}$ to forge a valid hash collision is negligible.

E. Design Goals

The macro aim of this article is to design a novel blockchain architecture which is compatible with IoT-based edge computing systems. To overcome the storage constraint barrier, we aim to use the idea of a "redactable blockchain" to enable the substitution of the historical blocks without breaking the links between the original blocks. In this article, we aim to improve the security and efficiency of the substitution operation in addition to our previous work DTTCH scheme. The details of the micro design goals are as follows.

1) *Efficiency:* Compared to the ordinary block generation (below the storage limit of the edge devices), the proposed scheme should have a close performance during the block substitution process.
2) *Security:* The proposed scheme should have fundamental security properties, including indistinguishability, collision resistance, and robustness. In addition, it should also guarantee the trapdoor key's security, with entities' shares (less than t), the adversaries can not reconstruct the original trapdoor key to find hash collisions.

V. FS Substitution Scheme

In this section, we first present the FS substitution scheme to address the security vulnerabilities caused by the exposure of randomness R in the DTTCH scheme. In addition, we introduce an optimized method by integrating fast Lagrange interpolation and FFT. Finally, we discuss the detailed process of block substitution.

A. Construction of FS Scheme

To hide the randomness R, inspired by Sigma Protocol [23], instead of using zero-knowledge proof primitives, we construct a lightweight FS scheme without sacrificing the zero-knowledge nature but improving the security compared to the DTTCH scheme.

1) *FS.Setup* $(1^\lambda) \rightarrow$ *(params):* Choose a random oracle $H' : \mathbb{G}_T \rightarrow \mathbb{Z}_p$. The algorithm **FS.Setup**$(\cdot)$ takes as input a security parameter 1^λ and outputs the system parameter $params = (\mathbb{G}_1, \mathbb{G}_2, p, g_1, g_2, H')$, g_1, g_2 is the generator of $\mathbb{G}_1$, $\mathbb{G}_2$, respectively.
2) *FS.Gen (params, t, n)* $\rightarrow$ *(hk,* $(sk_1, \ldots, sk_n)$*,* $(pk_1, \ldots, pk_n)$*):* The algorithm takes as input the system parameter *params*, a predefined threshold t and the total number of the entities in the system n, chooses a random number $x \xleftarrow{R} \mathbb{Z}_p$ and $\widehat{h} \xleftarrow{R} \mathbb{G}_1$, and sets $h_1 = g_1^x$ and $h_2 = g_2^x$, then computes d s.t. $xd \equiv 1 \bmod p$, picks a polynomial v of degree $t-1$ with coefficients from $\mathbb{Z}_p$, and sets the constant term of v to d (which means $v(0) = d$). Finally, the algorithm outputs n secret keys $sk_i = v(i)$, public keys $pk_i = g_2^{sk_i}$ and the public hash key $hk = (h_1, h_2, \widehat{h})$.
3) *FS.Hash (m, hk)* $\rightarrow$ $(\hbar, \pi)$*:* The algorithm takes as input a message m and a hash key hk, chooses a random number $r \xleftarrow{R} \mathbb{Z}_p$, computes a hash $\hbar = h_1^r \widehat{h}^m$ and a randomness $R = g_1^r$. And then generates a random hidden parameter $z \xleftarrow{R} \mathbb{Z}_p$ and takes it as an input to further generate another hidden parameter $\zeta = H'(P)$, where $P = e(g_1, g_2^x)^z$. Hides R in the exponent as forms of Sigma protocols and obtains $w = R^\zeta / g_1^z = g_1^{\zeta r + z}$, denote $\pi = (w, P)$ as the proof of randomness R. The algorithm then outputs $(\hbar, \pi)$.
4) *FS.HVer* $(m, \hbar, hk, \pi) \rightarrow$ *(True or False):* The algorithm takes as input a committed message m, a hash $\hbar$, a hash key hk and a randomness proof π, and outputs $\zeta \leftarrow H'(P)$ through the public parameter H', then verifies

$$P \cdot e(w, h_2) = e\left(\left(\frac{\hbar}{\widehat{h}^m}\right)^\zeta, g_2\right) \tag{1}$$

if equals, it outputs *True*, otherwise it outputs *False*.

5) *FS.Sign* $(sk_i, m', \hbar) \rightarrow (\sigma_i)$*:* The algorithm takes as input a secret key sk_i, a message m' and the hash $\hbar$, and outputs the message m''s credential

$$\sigma_i = \left(\frac{\hbar}{\widehat{h^{m'}}}\right)^{sk_i}. \tag{2}$$

6) *FS.SVer* $(pk_i, \sigma_i, m', \hbar) \rightarrow$ *(True or False):* The algorithm takes as input public keys pk_i, credentials σ_i, a message m', a hash $\hbar$, verifies

$$e\left(\frac{\hbar}{\widehat{h^{m'}}}, pk_i\right) = e(\sigma_i, g_2) \tag{3}$$

if equals, it outputs *True*, otherwise it outputs *False*.

7) *FS.Hcol* $((\sigma_1, \ldots, \sigma_t), \pi, hk, \hbar, m) \rightarrow (\perp$ *or* $\pi')$*:* The algorithm takes as input t credentials $(\sigma_1,\ldots,\sigma_t)$, a randomness proof π, a public hash key hk, a hash $\hbar$, and a message m, the algorithm first verifies by running **FS**.**HVer**$(m, \hbar, hk, \pi)$, if verified, the algorithm returns $\perp$. Otherwise, it parses each σ_i for $i \in [1, \ldots, t]$ and outputs Lagrange coefficient

$$l_i = \left[\prod_{j=1, j\neq i}^{t} (0-j)\right]\left[\prod_{j=1, j\neq i}^{t} (i-j)\right]^{-1} \bmod \quad p. \tag{4}$$

The algorithm then computes a new randomness $R' = \prod_{i=1}^{t} \sigma_i^{l_i}$, and the new proof $\pi' = (w', P')$ of R' as described in **FS**.**Hash**$(\cdot)$. It next checks if the following equation holds:

$$P \cdot e(w, h_2) = e\left(\left(\frac{\hbar}{\widehat{h^m}}\right)^{\zeta}, g_2\right). \tag{5}$$

If yes, the algorithm outputs π', otherwise, it outputs $\perp$.

B. Optimized Construction

1) FS-I Optimization: In the FS scheme, it incurs massive computational overheads during the polynomial multiplication in the process of generating secret keys and computing the Lagrange coefficient. Given a t-degree polynomial ϕ and a set of points, $\{x_1, \ldots, x_n\}$, and a set of function values, $\{f_1, \ldots, f_n\}$, the current FS scheme requires $\mathcal{O}(nt)$ arithmetic operations. To improve the FS scheme's computational performance, inspired by Fast polynomial interpolation methods, we propose the optimized scheme, named FS-I, FS-I focuses on optimizing **HCol**$(\cdot)$, and the procedure is as follows.

1) Separately compute $\phi(x)$.
   Part(1): $\phi_L(x) = \phi(x) \bmod (x-x_1)(x-x_2)\cdots(x-x_{n/2})$.
   Part(2): $\phi_R(x) = \phi(x) \bmod (x - x_{n/2+1}))(x - x_{n/2+2}) \cdots (x - x_n)$.
2) Compute recursively and further differentiate Part(1) and Part(2).
3) Output the value of $\phi_L(x)$ from $x_1, x_2, \ldots, x_n/2$, and the value of $\phi_R(x)$ from $x_{n/2+1}, x_{n/2+2}, \ldots, x_n$.
4) Finally, get the value of the leaf node on the last level of recursion $\phi(x) \bmod (x - x_i)$, based on the Polynomial Reminder Theorem, the value equals $\phi(i)$.

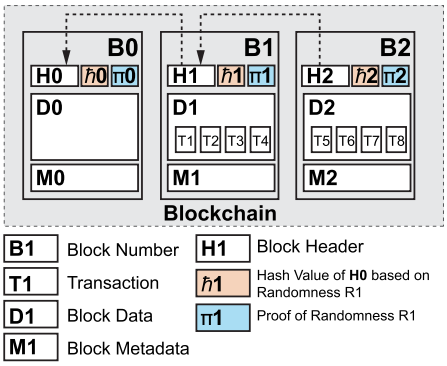

Fig. 2. Block structure.

To get the value of interpolation function $\mathcal{L}_i = \prod_{j\in T, j\neq i}[(x-j)/(i-j)]$ when $x = 0$, the FS-I scheme will first define $N(x) = \prod_{i\in T}(x - i)$, then get the numerator $N_i(x) = [N(x)]/[x - i] = \prod_{j\in T, j\neq i}(x-j)$, and the denominator $D_i = N_i(i) = \prod_j \in T, j \neq i(i - j)$, and we have

$$\mathcal{L}_i(x) = \frac{N_i(x)}{D_i}. \tag{6}$$

Therefore, through FS-I, it minimizes the time complexity from $\mathcal{O}(nt)$ to $\mathcal{O}(t\log^2 t)$ which dramatically improves the computational efficiency, especially, for $t > 64$.

2) FS-II Optimization: To further optimize the computational efficiency of Lagrange interpolation, we use FFT as a building block in the computation. Given a set of points $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$, there exists only one Lagrange polynomial ϕ whose degree is below n, $\phi = \sum_{i\in[n]} \mathcal{L}_i^{[n]}(x) y_i$, and the Lagrange interpolation

$$\mathcal{L}_i^{[n]}(x) = \prod_{j\in[n], j\neq i} \frac{x - x_j}{x_i - x_j}. \tag{7}$$

As $\mathcal{L}_i^{[n]}$ is determined by the set $\{x_i\}_{i\in[n]}$, by using FFT, the set will be transformed to $\{x_i\}_{i\in T}, T \subset [n]$, and $\{x_i\}_{i\in T}, T \subset [n]$.

In **Gen**$(\cdot)$ and **HCol**$(\cdot)$, FS-II will contribute to improving the computation of the elements in the finite cyclic group $\mathbb{Z}_p$. By replacing the entities id $\{1, \ldots, n\}$ with the unit-roots $\{\omega_n^{i-1}\}_i \in [n]$, it will significantly improve the performance because using unit root results in efficient modular inversion.

C. Block Substitution

1) Block Structure: Since each block substitution will generate a new randomness proof, and this proof must be included in the block for future verification, therefore, we propose a novel block structure for our proposed Hybrid nodes at the edge. As shown in Fig. 2, each block includes block headers, a hash of the previous block's header, the randomness R's proof π, transactions, and the block metadata. Each block is chained by including the hash of the previous block. Also, a block header includes a block number *BKNum*, a timestamp *TS*, a root hash *Root*, and a Merkle Hash Tree [24]. The *Root* is used for verifying the integrity of the data records. In Hybrid nodes, we use our FS scheme to generate the block header's hash and use the traditional hash (e.g., SHA265) for others.

TABLE II
SYSTEM INITIALIZATION

Initialization:

1 Key Authority generates secret keys $(sk_1, ..., sk_n)$, public keys $(pk_1, ..., pk_n)$ and the hash keys $hk = (h_1, h_2, \widehat{h})$;
2 Key Authority distributes $< sk_i, pk_i, hk >$ to $Node_i$.

2) Block Substitution Process: In the multilayer IoT-based smart grid system, the edge node is responsible for monitoring, controlling, and regulating the behavior of IoT devices based on their resource usage and activities. They collect sensing data among IoT devices along with resource management, utilize their own applications and resources to process, catch and upload data in real time, offload computation, and propose blocks through consensus. While the edge node storage reaches the maximum, the block substitution with hot data is executed for storing the ever-growing IoT data.

To execute the block substitution operation among the consortium of edge nodes, our proposed method can tolerate a considerable number of malicious or failure nodes. The block substitution will be executed by rounds, within each round, it will elect a node as the proposing node. However, a single Byzantine node may attack the network by imposing a large number of blocks. To address such venerability, we restrict each edge node can only propose one block per n rounds (n is the total number of the edge nodes in the network).

Before executing the block substitution operation, the Key authority in the system will first initialize by generating the system parameters, including threshold t and n pairs of public and secret keys $(sk_1, \ldots, sk_n)$, $(pk_1, \ldots, pk_n)$ and the hash key $hk = (h_1, h_2, \widehat{h})$ through $\mathbf{TT.Gen}(\cdot)$, and then assigns them to each edge node in the system. Note that under low-risk circumstances, a small threshold parameter t is sufficient as the loss may not be significant if an accident occurs; for medium-risk settings, a medium value of t is desirable to tolerate attacks for a high-risk environment, a high value of t might be required to tolerate failures, e.g., $t > n/2$. The detailed block substitution process is discussed as follows.

Step 1 (Block Proposal): During the block proposing phase, the system elects a leader node through$p = v \bmod |R|$, v represents the number of views, p is the leader node's ID, R is the total number of the edge nodes in the system. Once a leader is elected, it will collect all the transactions, package them into a block, and then broadcast them to the network. The detailed block proposing process is shown in Algorithm 1.

Step 2 (Voting Process): Once other edge nodes in the system receive *PROPOSAL* message, they will first conduct the verification process. As the public keys are public access and bonded with the transactions in the block. The edge nodes will use it to verify the signatures of d and Txs. The detailed voting process is shown in Algorithm 2.

Step 3 (Collision Finding): When the leader node received the credential from the rest of the nodes, it will first verify the credentials. Once the number of collected and verified credentials is sufficient(more than t), the leader node will then enter into the collision-finding process. After that, the leader will execute $\mathbf{FS.HCol}(\cdot)$ and generate the proof of randomness π. The detailed collision finding process is shown in Algorithm 3.

Step 4 (Block Confirmation): When the edge nodes received the message that contains *PROOF*, they will first verify the view number and the id of the proposed block, and then they will validate the proof of the randomness π. If both are verified, the edge nodes will update the block substitution pointer number to $R_{block} + 1$. Edge nodes will also send message *COMMIT* to the leader node and the full nodes separately. Message *COMMIT* includes the new block's header information. As full nodes have sufficient storage, they will append the new block to the last block on the chain after verification of the proofs. The detailed block confirmation process is shown in Algorithms 4 and 5.

Algorithm 1: Block Proposal

Input: Leader election results
Output: *PROPOSAL* message
Set $v = VIEW, n = R_{block} + 1$;
$Txs \leftarrow$ **GetTxFromTxPool**();
$MerkelRoot' \leftarrow$ **BuildMHT**(Txs);
Set $BlockHeader' = (n, TS', Root', PrevHash)$;
$d \leftarrow$ **Sig**($Hash(BlockHeader')$);
$Block \leftarrow$ **ConstructBlcok**($BlockHeader', \hbar, Txs, MetaData$);
Primary node $\rightarrow$ *All*: *PROPOSAL* message $<< PROPOSAL, v, n, d >, Block >$;

Algorithm 2: Voting Process

Input: *PROPOSAL* message
Output: Credential σ_i or Failure
For edge node N_i:
if $v == VIEW$ && $n == R_{block} + 1$ **then**
  if Sig_Verify(d)==*True* && **Tx_Verify**(Txs)==*True* && $\hbar ==$ **GetBlcok**(**n**).$\hbar$ **then**
    $\sigma_i =$ **FS.Sign**($sk_i, BlockHeader', \hbar$);
    $N_i \rightarrow Primary$:*VOTE* message $< VOTE, v, i, \sigma_i >$;
    N_i monitors the communication port to *Primary*;
  else
    return Failure ; // verification failure
  end
else
  return Failure ; // view ID and substitution pointer does not match.
end

VI. SECURITY ANALYSIS

A. Proof of Indistinguishability

Theorem 1: If the DCCTH scheme satisfies indistinguishability, the FS scheme will also hold.

Algorithm 3: Collision Finding Process

Input: *VOTE* message
Output: The proof of randomness π' or WAIT

For the leader node *Primary*:
for $\sigma_i, \quad i \in [1, N]$ **do**
  if FS.SVer$(pk_i, \sigma_i, BlockHeader', \hbar) == True$ **then**
    $ValidCredentialCount = ValidCredentialCount+1$;
  else
    DROP σ_i ; `// If verification fails, withdraw the credential`
  end
end
if $ValidCredentialCount \geq t$ **then**
  $\pi' \leftarrow$ **FS.HCol**$(\cdot)((\sigma_1, ..., \sigma_t), \pi, hk, BlockHeader')$;
  $sig \leftarrow Sig(\pi')$;
  $Primary \rightarrow All$: *PROOF* message
  $<< PROOF, v, n, \pi' >, sig >$;
else
  WAIT for ; `// If not received a sufficient number of credentials`
end

Algorithm 4: Confirmation Process

Input: *PROOF* message
Output: *COMMIT* message or Failure

if $v == VIEW \ \&\& \ n == R_{block} + 1$ **then**
  if Sig_Verify$(sig) == True \ \&\&$ **FS.HVer**$(BlockHeader', \hbar, hk, \pi') == True$ **then**
    Set $R_{block} = R_{block} + 1$;
    Substitute The Historical Block with the Proposal Block $N_i \rightarrow HC$: *Historical BlockHeader*
    Substitute and Backup BlockHeader
    $N_i \rightarrow Primary$: *COMMIT* message
    $< COMMIT, v, n, i, sig_i >$;
    $N_i \rightarrow Cloud$: *COMMIT* message
    $< COMMIT, BlockHeader', v, n, i, sig_i >$
  else
    return Failure;
  end
else
  return Failure;
end

Algorithm 5: Confirmation Process (Leader Node)

Input: *COMMIT* message
Output: Success or Failure

for $COMMIT \in N_i, \ i \in [1, N]$ **do**
  if Sig_Verify$(\mathbf{sig_i}) ==$ **True then**
    $CommitNum = CommitNum + 1$;
  else
    Pass ; `// If verification fails, it will not update the count`
  end
end
if $CommitNum \geq N/2 + 1$ **then**
  Substitute and Backup BlockHeader $Primary \rightarrow Fullnode$: $<BackupBlock, v, n, sig_{Primary}>$;
  return Success ; `// end Process.`
else
  return Failure;
end

Proof: The proof process is simulated by the simulator $\mathcal{S}$ and the adversary $\mathcal{A}$. Assume that the maximum number for $\mathcal{A}$ to execute the DCCTH algorithm is $n(\lambda)$. $\mathcal{S}$ acts as a distinguisher to attack DCCTH. $\mathcal{S}$ is able to obtain the hash public keys pk^*, and it is given oracle access to **Hash**$(\cdot)$ and **HCol**$(\cdot)$(denoted as**HashOrHCol**). While $\mathcal{A}$ can learn leaked messages from $\mathcal{S}$, its goal is to break the indistinguishability of DCCTH. $\mathcal{S}$ can randomly choose $g \in [1, n(\lambda)]$ as the index of executing **HashOrHCol**. At gth execution, $\mathcal{S}$ directly computes the hash $(\hbar, \pi) \leftarrow Hash(pk^*, m)$, but not to call the randomness $(\hbar, \pi)$ and the hash generated by **HCol**$(\cdot)$.

To set up the cryptographic game, $\mathcal{S}$ generates n nodes for $\mathcal{A}$. First, $\mathcal{S}$ random chooses a node from n, **Hash**$(\cdot)$ takes as input the chosen node's hash public key. $\mathcal{S}$ can access to the **Sign**$(\cdot)$ function of each node, and it can also obtain signed credentials from t out of n nodes. If $\mathcal{A}$ commits a set of messages (m_0, m_1) at gth execution. Although $\mathcal{S}$ can get the hash value $(\hbar_b, \pi_b)$ by calling the **HashOrHCol**, it can not distinguish whether the hash is generated by the **Hash**$(\cdot)$ function or the **HCol**$(\cdot)$ function. $\mathcal{S}$ outputs $(\hbar_b, \pi_b)$ and sends to $\mathcal{A}$. Since DCCTH has indistinguishability, $\mathcal{A}$ can not distinguish whether the randomness proof π_b is generated by **Hash**$(\cdot)$ or **HCol**$(\cdot)$, therefore, we have $|Pr(S_0) - Pr(S_1)| \leq \mathcal{V}_{DCCTH-ind(\lambda)}$. ■

B. Proof of Collision-Resistance

Theorem 2: If the Sigma protocol is witness-indistinguishable and zero-knowledge and satisfies the DCDH assumption, the FS scheme is Collision-Resistance.

Proof: We prove the FS's collision-resistance property by a series of cryptographic games in the following.

Game 0: This round is to initialization for the game. For each execution request, the adversary $\mathcal{A}$ calls **HCol**$(\cdot)$ from the challenger $\mathcal{C}$ to get $(\hbar, m', (\sigma_1, \ldots, \sigma_i))$, $\mathcal{C}$ computes $R' = \prod_{i=1}^{t} \sigma_i^{\mathcal{L}_i}$ and $w = R'^{\zeta}/g_1^z$ to output proof $\pi = (w, P)$.

If $\mathcal{A}$ forges proof $(\hbar^*, (m^*, \pi^*), (m'^*, \pi'^*))$, $\mathcal{C}$ will verify π^* and π'^* by

$$P^* \cdot e(\pi^*, h_2) = e\left(\left(\frac{\hbar^*}{\widehat{h^{m^*}}}\right)^{\zeta^*}, g_2\right) \tag{8}$$

$$P'^* \cdot e(\pi'^*, h_2) = e\left(\left(\frac{\hbar^*}{\widehat{h^{m'^*}}}\right)^{\zeta'^*}, g_2\right). \tag{9}$$

Game 1: Assume $\mathcal{A}$ tries to obtain the randomness r from **Hash**$(\cdot)$, and verifies through **HVer**$(\cdot)$. Since DL assumption holds, $\mathcal{A}$ can not get r from $w = g_1^{\zeta r+z}$ within *PPT* time.

Game 2: Assume $\mathcal{A}$ tries to forge proof π. As in Sigma protocol, proving indistinguishability can be reduced to prove a

general zero-knowledge property. So for (w_1, w_2) in (π_1, π_2), we should prove that $\mathcal{A}$ can not distinguishw_1 and w_2 within *PPT* time. Since $\mathcal{A}$ can not learn any knowledge from w^*, therefore, $\mathcal{A}$ can not get w^* within *PPT* time.

Game 3: Given DCDH tuple $(g_1, g_2, g_1^x, g_2^x, g_1^y)$ to the challenger $\mathcal{C}$, $\mathcal{A}$ sets the system parameters $(h_1 = g_1^x, h_2 = g_2^x, \widehat{h} = g^y)$. For $(\hbar^*, R^*, R'^*, m^*, m'^*)$ given by $\mathcal{C}$, $\mathcal{A}$ computes $hash(m^*, R^*) = hash(m'^*, R'*)$ and gets $g_1^{(y/x)} = (R'^*/R^*)^{(1/[m^*-m'^*])}$. If $\mathcal{A}$ sets $R' = g_1^{y/x}$ and forges R''s proof $\pi' = (w, P)$ $(w = R'^{\zeta}/g_1^{z'})$, $\mathcal{C}$ receives $(\hbar^*, \pi')$ and verifies $P \cdot e(w, h_2) = e((\hbar/\widehat{h^m})^{\zeta}, g_2)$ through $\mathbf{HVer}(\cdot)$

$$\begin{aligned}\text{LHS} &= e\left(g_1, g_2^x\right)^{z'} \cdot e\left(g_1^{(\frac{y}{x})\zeta - z'}, g_2^x\right) \\ &= e(g_1, g_2)^{xz'} \cdot e(g_1, g_2)^{(\frac{y}{x}\zeta - z')x} \\ &= e(g_1, g_2)^{y\zeta} \qquad (10)\end{aligned}$$

$$\begin{aligned}\text{RHS} &= e\left(\left(\frac{\hbar^*}{\widehat{h^{m^*}}}\right)^{\zeta}, g_2\right) = e\left(\left(\frac{h_1^{r'}\widehat{h^{m^*}}}{\widehat{h^{m^*}}}\right)^{\zeta}, g_2\right) \\ &= e\left(h_1^{r'\zeta}, g_2\right) = e\left(g_1^{xr'\zeta}, g_2\right) \\ &= e(g_1, g_2)^{xr'\zeta}. \qquad (11)\end{aligned}$$

From the above equations, we have LHS $\neq$ RHS, that is to say, $\mathcal{A}$ can not forge a valid proof through $(\hbar^*, m^*)$. ■

C. Proof of Robustness

Theorem 3: If the benign entities number$t > n/2$ in the system, and the DL assumption holds, the FS scheme is robust.

Proof: In the robustness experiment $Adv^{\text{ROB}}_{\mathcal{A},\text{TTCH}}(\lambda)$, assume that the maximum number of entities for the adversary $\mathcal{A}$ can control is $t-1$, that means $\mathcal{A}$ can get a maximum number of $t-1$ signatures. For $\mathcal{A}$ to succeed in the experiment, $\mathcal{A}$ must first sign the new messages m' with the $t-1$ secret keys, and forge a valid credential by satisfying the following equation.

1) $\mathbf{SVer}(\sigma_i, m'^*, pk_i, \hbar, hk) = \text{True}$.
2) $\sigma_i \neq \mathbf{Sign}(sk_i, m'^*, \hbar)$.

Since all entities' key pairs are generated by $\mathbf{Gen}(\cdot)$ during the system initialization, the forged credential share by $\mathcal{A}$ must be verified by the corresponding entity's public key. Since each valid credential σ_i is signed by the same entity's secret key, it implies that for a $\mathcal{A}$ to succeed, it must obtain one more entity key pair in addition to the $t-1$ key pairs. As $\mathcal{A}$ tries to obtain secret keys through monitoring the valid credentials sent by the entities, however, for $\mathcal{A}$ to be able to succeed, he must break the DL assumption. Since $\mathcal{A}$ can not break DL assumption within *PPT* time, therefore, the FS scheme is robust. ■

VII. Performance Evaluations

A. FS Related Experiments

We first conduct a series of comparison experiments to evaluate the performance of the DTTCH scheme and our proposed FS scheme.

Setup: We implement the construction of **FS** with C++ by using cryptographic library *libff* and polynomial library *libfqfft*. Moreover, the bilinear pairing is defined over the 256-bit Barreto-Naehrig [25] curve, the FFT is defined over *libfqfft*. The experiments are deployed on computing nodes with Intel Core i7-10700 CPU with 6 core processors running at 2.9 GHz and 16-GB RAM with Ubuntu 18.04 system.

TABLE III
Size of the Main System Parameters

Parameter	Size
Hash Key hk	96 byte
Secret Key sk_i	32 byte
Public Key pk_i	32 byte
Hash value $\hbar$	32 byte
Proof of Randomness π	32 byte
Credential σ_i	32 byte

TABLE IV
Performance of DTTCH Scheme (Threshold t Varies From 5 to 30, Unit: Millisecond)

Operation	t=5	t=10	t=15	t=20	t=25	t=30
$\mathbf{TT.Gen}(\cdot)$	4.67	5.24	6.07	7.34	8.96	10.98
$\mathbf{TT.Hash}(\cdot)$	1.02	1.01	1.03	1.02	1.02	1.01
$\mathbf{TT.HVer}(\cdot)$	4.99	5.04	5.04	5.01	5.00	5.02
$\mathbf{TT.Sign}(\cdot)$	3.76	6.97	10.48	13.91	17.28	20.86
$\mathbf{TT.Hcol}(\cdot)$	3.02	13.59	40.74	88.56	169.85	295.08

The default threshold t satisfies $t > n/2$. For simplicity, we set the size of the message to 1 kB, and each result is calculated by the average running time of 1000 executions. The experiments first initialize the ellipse curve through randomizing, and the functions will be computed based on the same ellipse curve.

The main system parameters are shown in Table III. The size of *hk* is larger than other parameters because *hk* includes three parts which are generated from $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{Z}_p$, respectively. Since *hk* is generated during the system initialization process, it will not affect the performance of the system afterwards.

The size of the symmetric bilinear groups and $\mathbb{G}_2$ used in this experiment are all 32 bytes, where the secret key is generated by $\mathbb{G}_1$, the public key is generated by $\mathbb{G}_2$. As we use a 256-bit Bn elliptic curve, the security is equivalent to the security of a 3072-bit RSA algorithm, but it generated a smaller hash and credentials than the same-level RSA algorithm. During the system initialization phase, key authority distributes hash keys, secret keys, and public keys to each entity in the system. Much message communication overheads will incur during the four phases of block substitution, therefore, using smaller parameters will decrease the communication costs.

Performance Evaluation: The performance is shown in Table IV, and the experimental performance results of the FS scheme are shown in Table V. The FS scheme optimizes the security and computational efficiency over the DTTCH scheme by introducing a novel $\mathbf{FS.SVer}(\cdot)$ to verify the credentials. In addition, the FS scheme uses the fast Lagrange interpolation method to find collisions while using FFT to calculate the polynomial. This experiment sets the number of entities $n = 2t - 1$, and records the time cost of six algorithms in the FS scheme with the threshold parameter t increasing from 8 to 512, where t is a geometric progression.

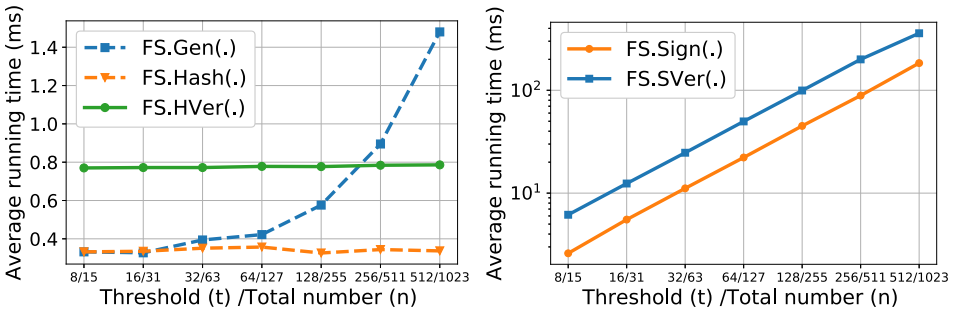

Fig. 3. Performance evaluation of FS scheme. (a) Computational costs of **FS**.**Gen**(·), **FS**.**Hash**(·), and **FS**.**HVer**(·). (b) Computational costs of **FS**.**Sign**(·) and **FS**.**SVer**(·).

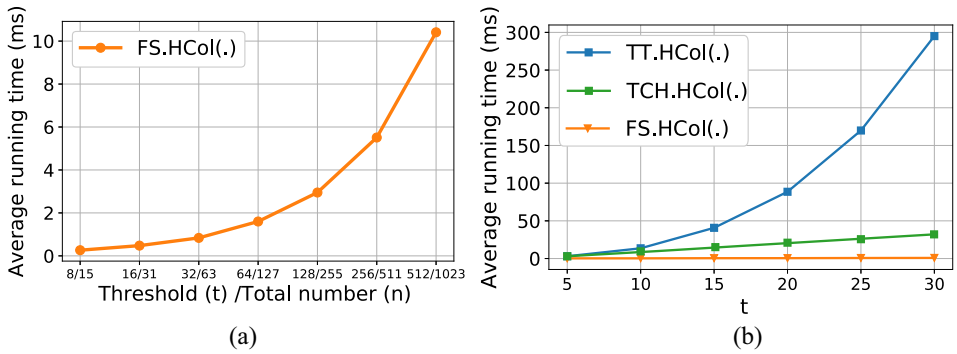

Fig. 4. Comparison experiments of **HCol**(·) in different schemes. (a) Computational costs of **FS**.**HCol**(·). (b) Performance comparison with **TT**.**HCol**(·) and **TCH**.**HCol**(·).

TABLE V
PERFORMANCE OF FS SCHEME (THRESHOLD t VARIES FROM 8 TO 512, UNIT: MILLISECOND)

Operation	t=8	t=16	t=32	t=64	t=128	t=256	t=512
FS.**Gen**(·)	0.33	0.32	0.39	0.42	0.57	0.89	1.48
FS.**Hash**(·)	0.33	0.33	0.35	0.35	0.32	0.34	0.33
FS.**HVer**(·)	0.77	0.77	0.77	0.78	0.77	0.78	0.78
FS.**Sign**(·)	2.6	5.53	11.13	22.20	45.02	88.83	183.79
FS.**SVer**(·)	6.16	12.40	24.70	49.79	99.46	199.72	358.40
FS.**Hcol**(·)	0.26	0.47	0.83	1.6	2.22	3.97	7.14

Fig. 3(a) evaluates the average performance overhead of the FS scheme for key generation, hash generation, and hash verification. Compared with the DTTCH scheme, FS is over ten times faster, it is because FS uses the unit root and FFT calculations to generate the selected polynomial secret keys during the key generation phase. Both hash and verification operations are single computational operations and their cost does not change with the increase of threshold t. In addition, as hash and pairing function verification involve polynomial calculations, the time cost required for both operations is also optimized. Fig. 3(b) shows the cost for signing and verification in FS. This experiment records the time costs of signing by all participants. Since only t valid certificates need to be collected in FS to calculate collisions, the average cost of verification is calculated by randomly selecting t certificates from all participant certificates in the experiment. It can be seen from Fig. 3(b) that the time cost to sign the certificate and verify it is linearly related to t. In real-world settings, the signing operation can be performed by all the participants in parallel. Although the verification operation can only be performed by a single participant in a serial manner, the results of this experiment show that the operation is still affordable in large-scale systems (greater than 100 entities).

Fig. 4(a) evaluates the performance of the **HCol**(·) with different threshold parameters. Secret-sharing algorithms in FS use fast Lagrange interpolation to aggregate t credentials on an exponential basis during collision computation, therefore, **HCol**(·) is significantly correlated to threshold parameters. As shown in Fig. 4(b), the performance of **FS**.**HCol**(·) has a significant raise compared to **TT**.**HCol**(·). **TT**.**HCol**(·) requires 300 ms to find collisions while **FS**.**HCol**(·) only takes 0.8 ms and only takes 11 ms when $t = 512$.

In addition, compared with the similar scheme Huang et al. [19], as shown in Fig. 4(b), the FS scheme does also exceed the performance of the TCH scheme proposed by Huang et al. As t increases, the costs of **TCH**.**HCol**(·) dramatically increases. Through the experiments, it demonstrates that the FS scheme is viable for large-scale IoT-based smart grid systems.

B. Rewrite Experiments

We instantiate a prototype of FS scheme-enabled blockchain with Go language. In addition, we also instantiate prototypes of DTTCH and TCH scheme-enabled blockchain as the comparison experiment. To evaluate the performance of the operations in the chain, we conduct a list of experiments on

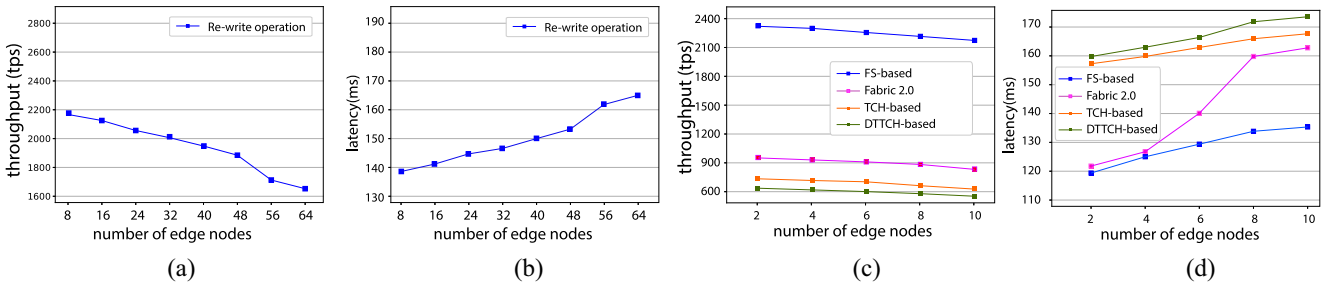

Fig. 5. Performance evaluation of the rewrite operation. (a) Throughput. (b) Latency. (c) Throughput comparison with baseline. (d) Latency comparison with baseline.

the rewrite operations including the comparison experiments with the baseline Hyperledger Fabric 2.0. We use PBFT in our prototype, however, in real settings, the consensus mechanism is plug-gable according to different requirements, for example, the network size, and other variables.

Setup: The prototype was deployed on the Aliyun CES as the experimental platform, each node is running in the Docker container on 20 physical edge nodes, and each server is equipped with Intel Core i7 10700@ 2.90 GHz, 16-GB RAM, the operating system is 64-bit Ubuntu 18.04.4 LTS and the network bandwidth is 10 Mb/s. Meanwhile, we use Raspberry Pi 4 Modele B to generate transactions and transmit them to the edge node. We use the Grpc protocol for communication between different nodes. We set the transactions per block to 300 because 300 is the performance bottleneck based on our past experience.

Experiment Evaluation-1: We use the number of nodes as the variable to evaluate the throughput and latency of the rewrite operations. We set the threshold to a fixed $t = N/2+1$, and increase the number of nodes from 8 to 64.

The throughput experiment results are shown in Fig. 5(a), which demonstrates that the throughput decreases slowly as the number of nodes increases. When there are eight nodes in the network, the throughput reaches 2166 transactions per second (TPS), while it decreases to around 1600 TPS when there are 64 nodes. It is reasonable because as the number of nodes increases, the network will incur more overheads, including communication costs, verification costs, and so on.

Experiment Evaluation-2: In addition, we also conduct the comparison experiment with the baseline (Hyperledger Fabric 2.0), DTTCH-based and TCH-based. We use the number of nodes as the variable to evaluate the throughput and latency of the rewrite operations of the comparison approaches. We set the threshold to a fixed $t = 5$, and increase the number of nodes from 2 to 10.

As shown in Fig. 5(c), the throughput of our approach is 2.5, 3, and 3.5 times faster than which of the baseline, DTTCH-based and TCH-based, respectively. It also implies that the performance will not lose much when there are fewer nodes in the network. For the latency experiments results, as shown in Fig. 5(b), the latency has an inverse relation with the throughput, the latency increases as the number of nodes increases. Moreover, as our experiments were done under an ideal environment, however, in real settings, the data generation rates of different IoT devices fluctuate, and the performance of our approach will also be affected.

VIII. Conclusion

In this article, we proposed FS to build a redactable blockchain for the edge. Through FS, we address the storage problem of traditional blockchain caused by ever-growing IoT data. The proposed block substitution operation is used when the edge node reaches the maximum storage, it empowers FS to substitute the historical blocks in a secure and controllable way while maintaining the connectivity of the chain. In addition, we optimize the FS scheme through fast polynomial interpolation and FFT to increase the performance by several magnitudes compared to which of DTTCH, while guaranteeing the security of randomness R at the same time. Through a detailed security analysis and comprehensive experiments, we demonstrate that our proposed methods are practical for a large-scale IoT-based smart grid setting. Finally, we will continue work on the arena of blockchain-enabled edge computing direction, hopefully, we will research the layer2 part for further optimizing the scalability of the blockchain.

References

[1] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1508–1532, 2nd Quart., 2019.

[2] L. Liu et al., "Blockchain-enabled secure data sharing scheme in mobile-edge computing: An asynchronous advantage actor–critic learning approach," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2342–2353, Feb. 2021.

[3] Y. Ju et al., "Joint secure offloading and resource allocation for vehicular edge computing network: A multi-agent deep reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, early access, Feb. 10, 2023, doi: 10.1109/TITS.2023.3242997.

[4] Y. Lu, Y. Qi, S. Qi, Y. Li, H. Song, and Y. Liu, "Say no to price discrimination: Decentralized and automated incentives for price auditing in ride-hailing services," *IEEE Trans. Mobile Comput.*, vol. 21, no. 2, pp. 663–680, Feb. 2022. [Online]. Available: https://doi.org/10.1109/TMC.2020.3008315

[5] M. B. Mollah et al., "Blockchain for future smart grid: A comprehensive survey," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 18–43, Jan. 2021.

[6] T.-V. Le, C.-L. Hsu, and W.-X. Chen, "A hybrid blockchain-based log management scheme with nonrepudiation for smart grids," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 5771–5782, Sep. 2022, doi: 10.1109/TII.2021.3136580.

[7] M. Li, C. Lal, M. Conti, and D. Hu, "LEChain: A blockchain-based lawful evidence management scheme for digital forensics," *Future Gener. Comput. Syst.*, vol. 115, pp. 406–420, Feb. 2021. [Online]. Available: https://doi.org/10.1016/j.future.2020.09.038

[8] R. de Best. "Bitcoin blockchain size 2010–2020." Jul. 2022. [Online]. Available: https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size

[9] V. Buterin. "A next generation smart contract and decentralized application platform." 2022. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[10] S. Xue, X. Zhao, X. Li, G. Zhang, and C. Xing, "A trusted system framework for electronic records management based on blockchain," in *Web Information Systems and Applications* (Lecture Notes in Computer Science 11817). Cham, Switzerland: Springer, 2019, pp. 548–559. [Online]. Available: https://doi.org/10.1007/978-3-030-30952-755
[11] H. Krawczyk and T. Rabin, "Chameleon signatures," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, San Diego, CA, USA, 2000, pp. 143–154.
[12] G. Ateniese and B. de Medeiros, "On the key exposure problem in chameleon hashes," in *Proc. 4th Int. Conf. Security Commun. Netw. (SCN)*, 2004, pp. 165–179.
[13] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, "Redactable blockchain—or—rewriting history in Bitcoin and friends," in *Proc. IEEE Eur. Symp. Security Privacy (EuroS P)*, 2017, pp. 111–126.
[14] J. V. Z. Gathen and J. Gerhard, "Fast polynomial evaluation and interpolation," in *Modern Computer Algebra*, 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2013, ch. 10.
[15] J.-P. Berrut and L. N. Trefethen, "Barycentric Lagrange interpolation," *SIAM Rev.*, vol. 46, no. 3, pp. 501–517, 2004.
[16] D. Derler, K. Samelin, D. Slamanig, and C. Striecks, "Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2019, pp. 18–21.
[17] D. Deuber, B. Magri, and S. A. K. Thyagarajan, "Redactable blockchain in the permissionless setting," in *Proc. IEEE Symp. Security Privacy (S P)*, May 2019, pp. 124–138.
[18] J. Zhang et al., "Serving at the edge: A redactable blockchain with fixed storage," in *Web Information Systems and Applications (WISA)* (Lecture Notes in Computer Science 12432), G. Wang, X. Lin, J. Hendler, W. Song, Z. Xu, and G. Liu, Eds. Cham, Switzerland: Springer, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-60029-7_58
[19] K. Huang et al., "Building redactable consortium blockchain for industrial Internet-of-Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3670–3679, Jun. 2019.
[20] Y. Lu et al., "Accelerating at the edge: A storage-elastic blockchain for latency-sensitive vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11862–11876, Aug. 2022. [Online]. Available: https://doi.org/10.1109/TITS.2021.3108052
[21] *Accenture Debuts Prototype of 'Editable' Blockchain for Enterprise and Permissioned Systems [EB]*, Accenture, Dublin, Ireland, Sep. 2016. Accessed: Aug. 10, 2017.
[22] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig, "Chameleon-hashes with ephemeral trapdoors," in *Proc. IACR Int. Workshop Public Key Cryptogr.*, 2017, pp. 152–182. [Online]. Available: https://doi.org/10.1007/978-3-662-54388-7_6
[23] C. P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991. [Online]. Available: https://doi.org/10.1007/BF00196725
[24] R. C. Merkle, "A certified digital signature," in *Proc. Annu. Int. Cryptol. Conf.*, 1989, pp. 218–238.
[25] K. Kasamatsu. "Barreto-Naehrig curves." 2014. Accessed: Aug. 14, 2014. [Online]. Available: https://tools.ietf.org/id/draft-kasamatsu-bncurves-01.html
[26] F. Bao, R. H. Deng, and H. Zhu, "Variations of Diffie-Hellman problem," in *Proc. Int. Conf. Inf. Commun. Security*, 2011, pp. 201–219.
[27] K. Ashritha, M. Sindhu, and K. V. Lakshmy, "Redactable blockchain using enhanced chameleon hash function," in *Proc. 5th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, 2019, pp. 323–328, doi: 10.1109/ICACCS.2019.8728524.
[28] J. Xu, K. Xue, H. Tian, J. Hong, D. S. L. Wei, and P. Hong, "An identity management and authentication scheme based on redactable blockchain for mobile networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6688–6698, Jun. 2020, doi: 10.1109/TVT.2020.2986041.
[29] K. Huang, X. Zhang, Y. Mu, F. Rezaeibagha, X. Du, and N. Guizani, "Achieving intelligent trust-layer for Internet-of-Things via self-redactable blockchain," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2677–2686, Apr. 2020, doi: 10.1109/TII.2019.2943331.

Youshui Lu (Member, IEEE) received the B.S. degree from The Australian National University, Canberra, ACT, Australia, in 2013, the M.S. degree from The University of Sydney, Camperdown, NSW, Australia, in 2015, and the Ph.D. degree from the School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China, in 2021.

He is currently an Assistant Professor with the School of Electrical Engineering, Xi'an Jiaotong University. His research interests include blockchain technology, distributed systems, Internet of Things, data security, and smart grid.

Xiaojun Tang (Member, IEEE) was born in Jiangxi, China, in 1973. He received the B.S. and M.S. degrees in control theory and control engineering from Xi'an University of Technology, Xi'an, China, in 1998 and 2001, respectively, and the Ph.D. degree in instrument science and technology from Xi'an Jiaotong University, Xi'an, in 2004.

From 2007 to 2008, he was a Postdoctoral Fellow with The University of New Orleans, New Orleans, LA, USA. Since 2014, he has been a Professor with the Department of Measurement and Control, School of Electrical Engineering, Xi'an Jiaotong University. His research interests include smart sensor and instrumentation, condition monitoring technology for power equipment, and smart control.

Lei Liu (Member, IEEE) received the B.Eng. degree in communication engineering from Zhengzhou University, Zhengzhou, China, in 2010, and the M.Sc. and Ph.D. degrees in communication engineering from Xidian University, Xi'an, China, in 2013 and 2019, respectively.

From 2013 to 2015, he worked with Technology Company. From 2018 to 2019, he was supported by China Scholarship Council to be a visiting Ph.D. student with the University of Oslo, Oslo, Norway. He is currently a Lecturer with the Department of Electrical Engineering and Computer Science, Xidian University. His research interests include vehicular ad hoc networks, intelligent transportation, mobile-edge computing, and Internet of Things.

F. Richard Yu (Fellow, IEEE) received the Ph.D. degree in electrical engineering from The University of British Columbia, Vancouver, BC, Canada, in 2003.

From 2002 to 2006, he was with Ericsson, Lund, Sweden, and a start-up in California, USA. He joined Carleton University, Ottawa, ON, Canada, in 2007, where he is currently a Professor. His research interests include connected/autonomous vehicles, security, artificial intelligence, blockchain, and wireless cyber–physical systems.

Dr. Yu received the IEEE TCGCC Best Journal Paper Award in 2019, the Distinguished Service Awards in 2016 and 2019, the Outstanding Leadership Award in 2013, the Carleton Research Achievement Awards in 2012 and 2021, the Ontario Early Researcher Award (formerly, the Premiers Research Excellence Award) in 2011, the Excellent Contribution Award at IEEE/IFIP TrustCom10, the Leadership Opportunity Fund Award from Canada Foundation of Innovation in 2009, and the Best Paper Awards at IEEE ICNC18, VTC17 Spring, ICC14, Globecom12, IEEE/IFIP TrustCom09, and International Conference on Networking 05. He serves on the editorial boards of several journals, including the Co-Editor-in-Chief for *Ad Hoc & Sensor Wireless Networks* and a Lead Series Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING. He has served as the technical program committee co-chair of numerous conferences. He has been named in the Clarivate Analytics list of Highly Cited Researchers in 2019 and 2020. He is an IEEE Distinguished Lecturer of both Vehicular Technology Society (VTS) and Communication Society. He is an elected member of the Board of Governors of the IEEE VTS and the Editor-in-Chief for IEEE VTS Mobile World Newsletter. He is a registered Professional Engineer in the province of Ontario, Canada, and a Fellow of IET and Engineering Institute of Canada.

Schahram Dustdar (Fellow, IEEE) received the Ph.D. degree in business informatics from the University of Linz, Linz, Austria, in 1992.

He is currently a Full Professor of Computer Science (informatics) with a focus on Internet technologies heading the Distributed Systems Group, TU Wien, Wein, Austria. He has been the Chairman of the Informatics Section of the Academia Europaea, since December 2016.

Prof. Dustdar was a recipient of the ACM Distinguished Scientist Award in 2009 and the IBM Faculty Award in 2012. He is an Associate Editor for the IEEE TRANSACTIONS ON SERVICES COMPUTING, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology*. He is on the Editorial Board of IEEE. He has been a member of the IEEE Conference Activities Committee since 2016, the Section Committee of Informatics of the Academia Europaea since 2015, and the Academia Europaea: The Academy of Europe, Informatics Section since 2013.