This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2021.3062348, IEEE Transactions on Emerging Topics in Computing

Blockchain-based Edge Computing Data Storage Protocol Under Simplified Group Signature

Zhiwei Wang

***Abstract*—In contrast to the traditional cloud-based IoT structure, which imposes high computation and storage demands on the central cloud server, edge computing can process data at the network edge. In this paper, we present an edge computing data storage protocol employing the blockchain and the group signature, where the blockchain works as the trusted third party, offering a convenient platform for the data storage and protection, and the group signature is utilized for privacy identity information protection. Unlike the common group signature scheme, we propose a notion of *simplified group signature*, which removes the traceable property to improve the computational efficiency. In edge computing, the edge devices and the end devices can easily set up a long-term trust relationship through the cloud services, so the traceability may be of little utility in this scenario. In our protocol, we also design a new data validation scheme that has a proof size of only $O(1)$.**

***Index Terms*—simplified group signature; blockchain; data storage; privacy protection; edge computing**

I. Introduction

The Internet of Things (IoT) makes use of data collected from end devices to optimize the control of the world in many applications [1], [2], [3], such as vehicles, agriculture, and healthcare. However, IoT applications also introduce many security challenges along with their benefits. In the IoT systems, a malicious instruction may cause severe damage to the physical world, since the cyber world and physical world are tightly coupled [7], [8]. Security issues have been one of the most important barriers that prevent the wide adoption of IoT applications. However, designing efficient secure services for the IoT systems is a challenge since the IoT devices usually have low computational power and are not capable of executing complex computations. Edge computing is one alternative way to help solve this problem [4], [5], [6]. Unlike cloud computing, edge computing processes data at the network edge. The edge computing-based IoT architecture has three layers: things layer, edge layer and cloud layer [7]. The things layer is responsible for collecting data and controlling the physical world. Most of the end devices in the things layer are constrained in terms of energy, computational power and storage. Thus, those devices have different operation systems and use various low-power communication protocols. The edge layer is utilized to help the nearby end devices. Heavy computational loads can be offloaded to the edge devices, and communication heterogeneity can be masked by the edge devices. The edge devices can also help to manage the nearby end devices. The cloud layer is used to store and process the collected data, and it also provides support to many IoT applications.

Generally, there are several advantages of edge computing compared with cloud computing [7]. First, the edge devices usually have more resources than the end devices, and thus, some computation-intensive operations can be offloaded to the edge device, such as some ”expensive” cryptographic operations. Second, the edge devices can easiliy setup a long-term trust relationships with the nearby end devices through the cloud services as in [8]. Compared with the nearby but temporary connected edge devices, the permanent available cloud services are more trustable to the end devices. With the level trust, the cloud can issue verifiable credentials to the edge devices which is helpful for the end devices to setup a trust relationships with the edge device. In [8], Sha et al. discuss the four steps authentication scheme in detail. Once the trust relationships are setup, the frequently contacts between the edge devices and the nearby end devices make such trust relationships to be long-term and stable. Third, the edge devices can be used to protect the privacy information of end devices (users) by applying some cryptographic algorithms. Fourth, since the edge devices are close to the end devices, better real-time performance for many applications can be achieved. Fifth, since the edge devices usually have more information than the end devices, it is possible to deploy some optimized schemes on the edge devices.

To further protect the collected data, the convenient platform of blockchain [9], [10], [16] can be deployed between the cloud layer and the edge layer and works as a trusted third party, as shown in Fig. 1. In a blockchain, there are some users called miners working cooperatively to validate the transactions and generate blocks. The early version of blockchain has been utilized as cryptocurrency [11], such as Bitcoin or Ethereum. If the blockchain is used as a platform to serve the edge computing data storage and protection, the transactions are different from the cryptocurrencies. The transactions in this scenario are the requests sent from the end devices for data storing and accessing [16]. The authentication of these requests is addressed by the blockchain miners instead of a trusted third server. Thus, the collected requests can be stored in different addresses, and a user can find the storage addresses through the blockchain. The access control is determined by the majority of the blockchain miners without a centralized trusted server, which can avoid unauthorized accesses. All the

Zhiwei Wang is with Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, School of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China; and also with the JiangSu Provincial Key Laboratory of Computer Network Technology (e-mail: zhwwang@njupt.edu.cn).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2021.3062348, IEEE Transactions on Emerging Topics in Computing

activities such as accessing or modifying are all recorded by the blockchain, and any malicious attempts can be detected.

Moreover, when the end devices request to store or access the collected data, the edge device helps to create a transaction for these end devices, and a valid transaction should include a signature by the corresponding end device. If we use a common signature scheme, the verifying process should utilize the signers' identities or public keys to validate the signatures. However, in some IoT applications with higher privacy protection requirements [8], [19] such as smart health, the signers' identities should be preserved [1]. To solve this problem, group signature is a good cryptographic tool for the privacy-preserving process, ensuring anonymity, authentication, and accountability [20]. Group signature enables members of a group to generate a signature on behalf of the group, and it cannot be traced back to its signer, except for a special opening party, which can open any valid group signatures.

In edge computing, the edge devices are responsible for not only carrying out the computation-intensive tasks but also managing the nearby end devices, and thus, the edge devices are the natural group manager in a group signature scheme. Moreover, regarding the discussion above, the edge devices can easily set up long-term trust relationships with the nearby end devices by the cloud services, so it does not require the traceable property. Removing the traceability from a group signature scheme can further improve the computational efficiency. On the other hand, when the edge device helps the end devices to put the data into the storage cloud, it should provide a proof to the end devices that their data have been well stored. Thus, an efficient edge computing data storage protocol also requires an efficient storage data validating scheme.

Motivation. Our motivation is to propose an efficient and privacy-protecting blockchain-based edge computing data protocol, which utilizes the blockchain to be the trusted third party for ensuring the security of data storage and access. For the user's privacy protection, group signature is used as a foundation of the protocol. To improve the efficiency, we choose Clarisse et al.'s short group signature scheme [34] for designing the protocol since its signature size and computational cost are the best of the schemes in the standard model. Moreover, the efficiency of Clarisse et al.'s short group signature can be further improved by removing the traceable property. The data validation proof scheme also needs to be redesigned since the current Merkel tree scheme still requires a logarithmic-size proof.

Contributions. The contributions of this paper are listed as follows. 1) We first propose a definition and a security model of *simplified group signature* without the traceable property. We construct a concrete scheme by simplifying Clarisse et al.'s short group signature scheme [34]. Our scheme inherits the short signature size of Clarisse et al.'s scheme and further decreases the computational cost of their scheme. 2) We design a *blockchain-based edge computing data storage protocol* under the simplified group signature scheme. The protocol utilizes the blockchain for ensuring the security of data access and uses the simplified group signature scheme to protect the users' privacy identity information. In an IoT application, data can be stored in the distributed databases while the corresponding access control list (ACL) is stored in the blockchain. When a user requests data from databases, the blockchain will decide whether the access is valid or not, and the decision works are handled by the blockchain miners. 3) In the proposed protocol, we design a new data validation scheme with a proof size of only $O(1)$.

Organization. The related works are described in Section II, and the security definition of the simplified group signature and a concrete scheme are proposed in Section III. In this section, we also provide a security proof for our scheme. Section IV presents a blockchain-based edge computing data storage protocol. In Section V, we discuss the security properties of our protocol. Section VI discusses the performances of our protocol over the platform of MacBook Pro and Edison. Finally, we conclude our paper in Section VII.

II. Related Works

To achieve secure authentication, privacy protection and collaborative sharing in IoT applications, there are many proposals of combining blockchain and edge computing techniques. Cui et al. proposed a new decentralized and trusted platform for edge computing, which provides a unified interface to the end devices, resolves the end device's compuational requests to the nearby edge device through a domain name server, and returns the computational results to the end device [12]. To incent the edge devices, their platform integrates the blockchain technology with edge computing, such that the contributions of each edge device could be accounted and rewarded. Chuang et al. proposed a trust-aware IoT data economic system with complete IoT data pricing, trading and protection functions [13]. They utilizes a multi-access architecture of edge computing to lighten the heavy tasks of end devices from blockchain operations, and reduce the trading latency. However, the above proposals have no relations with the cryptographic primitives.

Guo et al. proposed a trusted authentication platform combining edge computing and blockchain for realizing secure authentication and data sharing among different IoT applications [14]. In their platform, a distributed authentication mechanism based on elliptic curve cryptography (ECC) is presented. Ma et al. proposed a blockchain-based trusted data management scheme in edge computing [15], which supports multichannel data segment and privacy protection of sensitive data. In their scheme, they design a user-defined sensitive data encryption mechanism based the certificate-based encryption algorithm. Li et al. proposed a blockchain-based IoT data storage and protecting protocol [16]. They utilize the certificateless signature [21], [22] for verifying the transactions in the blockchain. All the above proposals cannot achieve the signer's anonymity, since the public key (certificate-based) or the identity of the signer should be used to verify his signature. Wang et al. proposed a blockchain-based mutual authentication and key agreement protocol for edge computing [17]. They use the dynamic generated public key instead of the real identity for

[1]In smart health, the identity information of patients or doctors is usually required to be preserved.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2021.3062348, IEEE Transactions on Emerging Topics in Computing

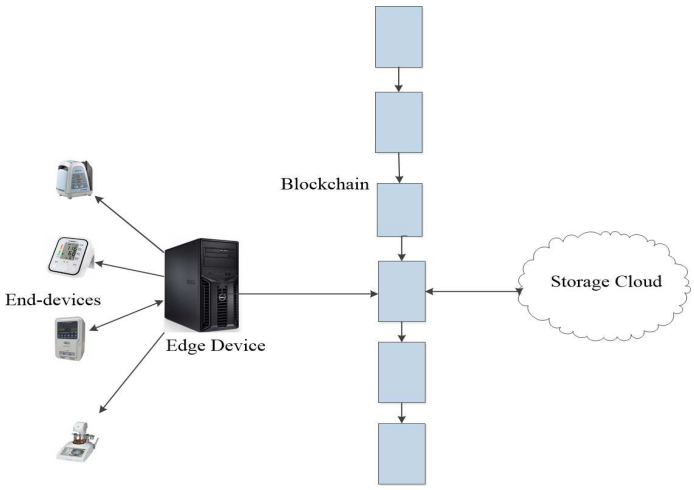

Fig. 1: Blockchain Used for an Edge Computing System

the authentication, which can achieve the identity anonymity. Enrnest et al. proposed a privacy-enhancement scheme that preserves the privacy details of the user in a environment combining edge computing and blockchain [18]. Their idea is similar to Wang et al's protocol, which use the randomly generated public keys instead of the real identity. Although the one-time (dynamic) public keys can hide the real identity or the long-term public key (which also indicate the signer's identity), a proof of zero-knowledge should be provided that the one-time public keys are generated from the real identity or the long-term public key, so that the correponding privacy scheme can be convinced. Unfortunately, both Wang et al's protocol [17] and Enrnest et al.'s scheme [18] do not provide such proofs.

Group signature may be an alternative method for solving the privacy problem. Chaum et al. first presented the notion of group signature, which enables members of a group to sign on behalf of the group [20]. The first practical group signature scheme was provided by Ateniese et al. [23], which combines seemingly contradictory properties: anonymity and authentication. A few years later, Bellare et al. proposed the first security model for the static group signature [24], which was later extended to the dynamic group signature [25]. Following this progress, the issue of designing an efficient group signature scheme became a great challenge. Bichsel et al. proposed an efficient scheme at the cost of a slightly weaker notion of anonymity [26]. Their construction can be further improved with the randomizable scheme of Pointcheval et al. [27]. Derler et al. proposed another group signature scheme [31] based on an equivalence-class signature [32], which achieves full anonymity at the cost of increased complexity. However, all of the above schemes are proved in the random oracles. Gai et al. directly utilize Boneh et al.'s group signature scheme [28] in their permissioned blockchain edge model for privacy protection [29], by combining blockchain and edge computing techniques. In their model [29], the group signature scheme and the covert channel authorization techniques are used to guarantee end devices' validity. However, Boneh et al.'s group signature scheme is also proved in the random oracle model by using Bellare et al.'s security definition for group signatures [24]. Zhang et al. proposed a modified group signature scheme for blockchain-based mobile-edge computing architecture [30]. However, their group signature scheme is only designed for the aggregation, and the most important property - anonymity is eliminated. Very recently, Backes et al. proposed a different framework based on signatures with flexible public keys, which yields construction in the standard model [33]. However, the computational cost of their schemes is still three times larger than the computational cost of the most efficient scheme in the random oracles. After that approach, Clarisse et al. proposed a short group signature scheme that can be proved secure in the standard model [34], and its size and computational cost are both the most efficient in the standard model. Furthermore, the authors' construction is also competitive against the most efficient scheme in the random oracles.

In edge computing data storage, there is another problem: when the edge device helps the end device forward its data set to the cloud, the end device will request a proof that its data have been well stored. The first simple solution is that the end device keeps the hash value of its entire data set $H(item_1, \cdots, item_n)$, and the edge device returns all items of the data set to be validated. This approach requires $O(n)$ network traffic for validating a single item. The second solution is the Merkel tree scheme [35], [36]. If the end device wants to inquire about the item d in the data set, then the edge device sends it the item d and a logarithmic-size proof. The end device computes a function of the item and the proof and compares the result to the hash value that it has saved (Merkel root). For example, in Fig. 2, given the item d and H_1,H_2,H_3 hash values, the end device can verify $H(H_3||\cdot) \stackrel{?}{=} H((H(d||H_1)||H_2)||H_3)$ to validate the item d.

III. Simplified Group Signature

A. Preliminaries

We first review the definition of the type 3 bilinear pairing groups: G_1, G_2 and G_T with the prime order p. We define e :

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2021.3062348, IEEE Transactions on Emerging Topics in Computing

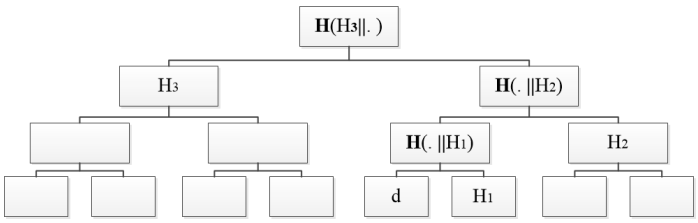

Fig. 2: Merkle Tree

$G_1 \times G_2 \to G_T$ to be the bilinear map that has the following properties:

1) Bilinear: $\forall g \in G_1, \hat{g} \in G_2, a, b \in Z_p$, $e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab}$.
2) Nondegenerate: $\exists g \in G_1, \hat{g} \in G_2$, $e(g, \hat{g}) \neq 1_{G_T}$.
3) Efficient Computability: There exists an efficient algorithm to compute $e(g, \hat{g})$ for all $g \in G_1$ and $\hat{g} \in G_2$.

Pointcheval et al. proposed a randomizable signature (PS signature) scheme [27] that enables deducing a rerandomized signature σ' from any valid signature σ. The special characteristic of their scheme is that any one cannot link σ' and σ without knowing the corresponding message M. We review the **PS signature** in the following.

Setup(1^λ): This algorithm takes a security parameter λ as input and generates the type 3 bilinear group generator (G_1, G_2, G_T, e) with prime order p. It also chooses the generators $g \in G_1$ and $\hat{g} \in G_2$. It sets $X = g_x$ and $\hat{X} = \hat{g}^x$ for a random $x \in Z_p$. The public parameters are

$$PP = (G_1, G_2, G_T, e, g, \hat{g}, X, \hat{X}).$$

KeyGen(PP): This algorithm randomly chooses $y \in Z_p$ and computes the secret/public key pair as $(sk, pk) = (g^y, \hat{Y} = \hat{g}^y)$.

Sign(sk, M): This algorithm randomly chooses $r \in Z_p$ and generates a signature as $(\sigma_1, \sigma_2) = (g^r, X^r \cdot g^{r \cdot y \cdot M})$.

Verify($pk, (\sigma_1, \sigma_2), M$): This algorithm outputs 1 if the following equation holds:

$$e(\sigma_1, \hat{X} \cdot \hat{Y}^M) = e(\sigma_2, \hat{g}),$$

which means that the signature (σ_1, σ_2) is valid; otherwise, it outputs 0.

An interesting feature of the PS signature is that anyone can rerandomize a valid signature by raising (σ_1, σ_2) to the same power.

Fuchsbauer et al. presented an equivalence-class signature (**FHS signature**) [37] for the equivalence relation $(M_1, \cdots, M_n) \sim (M_1^a, \cdots, M_n^a)$, where $a \in Z_p$. For the simplified group signature, we only consider the case $n = 1$, so we define the **FHS signature** scheme under this setting.

Setup(1^λ): This algorithm takes a security parameter λ as input and generates the type 3 bilinear group generator (G_1, G_2, G_T, e) with prime order p. It also chooses the generators $g \in G_1$ and $\hat{g} \in G_2$. The public parameters are

$$PP = (G_1, G_2, G_T, e, g, \hat{g}).$$

KeyGen(PP): This algorithm randomly chooses $\alpha \in Z_p$ and computes the secret/public key pair as $(sk, pk) = (\alpha, \hat{A} = \hat{g}^\alpha)$.

Sign(sk, M): To sign a message $M \in G_1$ of some equivalence class, this algorithm randomly chooses $t \in Z_p$ and generates a signature as $(\tau_1, \tau_2, \hat{\tau}) = ((M^\alpha)^t, g^{1/t}, \hat{g}^{1/t})$.

Verify($pk, (\tau_1, \tau_2, \hat{\tau}), M$): This algorithm outputs 1 if the following equations hold:

$$e(\tau_1, \hat{\tau}) = e(M, \hat{A}), e(\tau_2, \hat{g}) = e(g, \hat{\tau}),$$

which means that the signature $(\tau_1, \tau_2, \hat{\tau})$ is valid; otherwise, it outputs 0.

Note that anyone can derive an valid signature on other representative M^r of the same equivalence class by selecting a random $t' \in Z_p$ and computing $(\tau_1^{r \cdot t'}, \tau_2^{1/t'}, \hat{\tau}^{1/t'})$.

B. Security model of Simplified Group Signature

A common group signature scheme usually involves three types of entities: a group manager, an opening authority and users. However, in our simplified group signature, there are only two types of entities since the traceable property has been removed, namely, a group manager and users, and it does not require an opening authority. A simplified group signature scheme only consists of five algorithms as follows.

Setup(1^λ): This algorithm takes a security parameter λ as input and outputs the public parameters PP of the system.

GKeyGen(PP): This algorithm takes the public parameters PP as input and outputs the group manager's master/public key pair (gsk, gpk).

Join(): This is an interactive protocol between the group manager and a user i who wants to join the group. At the end of the protocol, the user i obtains his or her group signing key usk_i.

Sign(usk_i, M): This algorithm takes a group signing key usk_i and a message M as input and outputs a group signature σ on M.

Verify(gpk, σ, M): This algorithm takes a group public key gpk, a group signature σ and a message M and outputs a bit $b \in \{0, 1\}$.

A common group signature should satisfy four security properties: *correctness*, *anonymity*, *traceability* and *non-frameability*. The property of correctness means that any user who has joined the group should be able to generate any valid signature σ on any message M. Anonymity requires that the group signature should be anonymous. Traceability requires that no one can produce a valid signature that cannot be traced back by an opening authority. Non-frameability means that anyone who is falsely accused of have generated a group signature can be recognized by an opening authority. However, in a simplified group signature, it does not require traceability and non-frameability since we assume that the group manager and the users have set up a long-term trust relationship. The game of anonymity should make use of the following oracles:

- $\mathcal{JO}$ is an oracle that runs at the user's side of the `Join` algorithm. It can be used by an attacker $\mathcal{A}$ acting as a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2021.3062348, IEEE Transactions on Emerging Topics in Computing

corrupted group manager. It returns the result regarding whether a user i has already joined the group.
- $\mathcal{CO}$ is an oracle that returns a group signing key usk_i of the user i, and the user i is said to be corrupted.
- $\mathcal{SO}$ is an oracle that returns $\texttt{Sign}(usk_i, M)$ if i has already joined the group.

The anonymity of a simplified group signature scheme can be defined as the following game.

Setup: The challenger $\mathcal{C}$ runs the `Setup` algorithm and sends public parameters PP to the attacker $\mathcal{A}$. Then, $\mathcal{C}$ runs the `GKeyGen` algorithm and sends the group public key gpk to the attacker $\mathcal{A}$.

Query: The following queries can be issued by $\mathcal{A}$.

$\mathcal{JO}$ Query: $\mathcal{A}$ can access $\mathcal{JO}$ alternatively and obtain the result regarding whether the user i has joined the group.

$\mathcal{CO}$ Query: $\mathcal{A}$ can access $\mathcal{CO}$ for polynomial times and obtain the group signing keys of users.

$\mathcal{SO}$ Query: $\mathcal{A}$ can access $\mathcal{SO}$ adaptively and obtain the corresponding group signatures.

Challenge: The attacker $\mathcal{A}$ sends the indices of two honest users i_1 and i_0, and a message M^* to $\mathcal{C}$. $\mathcal{C}$ chooses a random bit b and returns a group signature σ^* on M^* using usk_{i_b}.

Output: The attacker $\mathcal{A}$ outputs a guess b' for b.

$\mathcal{A}$ wins the above game if $b' = b$.

We define $ADV_{\mathcal{A}}^{anon} = |Pr[\mathcal{A} \quad wins] - 1/2|$.

Definition 1: A simplified group signature scheme is anonymous if for any PPT attacker $\mathcal{A}$, $ADV_{\mathcal{A}}^{anon}$ is negligible.

C. Construction of Simplified Group Signature

In this section, we construct a new group signature scheme without traceability. Our scheme is a simplified version of Clarisse et al.'s short signature [34], which can be described as follows.

Setup(1^λ): This algorithm takes a security parameter λ as input, and generates the type 3 bilinear group generator (G_1, G_2, G_T, e) with prime order p. It also chooses the generators $g \in G_1$ and $\hat{g} \in G_2$. It sets $X = g_x$ and $\hat{X} = \hat{g}^x$ for some random $x \in Z_p$. Let $h : \{0,1\}^* \to Z_p$; the public parameters are

$$PP = (G_1, G_2, G_T, e, g, \hat{g}, X, \hat{X}, h).$$

GKeyGen(PP): The group manager chooses a random $\alpha \in Z_p$ and computes $(\hat{A}, \hat{B}) = (\hat{g}^\alpha, \hat{X}^\alpha)$. Then, he or she sets the group master key as $gsk = \alpha$ and the group public key as $gpk = (\hat{A}, \hat{B})$.

Join(): For joining the group, a user i chooses a random $u \in Z_p$ and sends g^u to the group manager. After receiving g^u, the group manager selects $t \in Z_p$ and returns $\upsilon_1' = (g^u)^{\alpha \cdot t}$, $\upsilon_2 = g^{1/t}$ and $\hat{\upsilon} = \hat{g}^{1/t}$ to the user i. Then, the user i computes $\upsilon_1 = (\upsilon_1')^{1/u}$ and sets his or her secret key as $usk_i = (\upsilon_1, \upsilon_2, \hat{\upsilon})$.

Sign(usk_i, M): For signing a message M, the user i chooses two random $s, r \in Z_p$, and computes $\upsilon_1' = \upsilon_1^{s \cdot r}$, $\upsilon_2' = \upsilon_2^{1/s}$, $\hat{\upsilon}' = \hat{\upsilon}^{1/s}$, $\sigma_1 = g^r$ and $\sigma_2 = X^{r/h(\upsilon_1' || \upsilon_2' || \hat{\upsilon}' || \sigma_1 || M)}$. The group signature is $\sigma = (\upsilon_1', \upsilon_2', \hat{\upsilon}', \sigma_1, \sigma_2)$.

Verify(gpk, σ, M): To verify the signature σ on message M, one checks whether the following equations hold:

$$e(\sigma_1 \sigma_2, \hat{A}) e(\sigma_1, \hat{B}^{-1/h(\upsilon_1' || \upsilon_2' || \hat{\upsilon}' || \sigma_1 || M)}) = e(\upsilon_1', \hat{\upsilon}'),$$

$$e(\upsilon_2', \hat{g}) = e(g, \hat{\upsilon}').$$

If one of the above equations is not satisfied, then one returns 0; otherwise, one returns 1.

Correctness.: Note that in the `Join` phase, the user's secret key $usk_i = (\upsilon_1, \upsilon_2, \hat{\upsilon})$ is also an FHS signature on g, which can be verified as $e(\upsilon_1, \hat{\upsilon}) = e(g, \hat{A})$ and $e(\upsilon_2, \hat{g}) = e(g, \hat{\upsilon})$.

Then, in the `Sign` phase, to generate a group signature on M, the user rerandomizes usk_i by using s, and the representative is updated to g^r; thus, the $(\upsilon_1', \upsilon_2', \hat{\upsilon}')$ is still an FHS signature that can be verified as $e(\upsilon_1', \hat{\upsilon}') = e(g^r, \hat{A})$ and $e(\upsilon_2', \hat{g}) = e(g, \hat{\upsilon})$. The following two elements (σ_1, σ_2) in σ comprise a PS signature on $M' = h(\upsilon_1' || \upsilon_2' || \hat{\upsilon}' || \sigma_1 || M))$ by using the random r. Therefore, the group's signature satisfies:

$$\begin{aligned}
& e(\sigma_1 \sigma_2, \hat{A}) e(\sigma_1, \hat{B}^{-1/h(\upsilon_1' || \upsilon_2' || \hat{\upsilon}' || \sigma_1 || M)}) \\
= \; & e(g^r \cdot X^{r/h(\upsilon_1' || \upsilon_2' || \hat{\upsilon}' || \sigma_1 || M)}, \hat{g}^\alpha) \\
& \cdot e(g^r, (\hat{X}^\alpha)^{-1/h(\upsilon_1' || \upsilon_2' || \hat{\upsilon}' || \sigma_1 || M)}) \\
= \; & e(g, \hat{g})^{\alpha r (1 + x/h(\upsilon_1' || \upsilon_2' || \hat{\upsilon}' || \sigma_1 || M))} \cdot e(g, \hat{g})^{-\alpha r x / h(\upsilon_1' || \upsilon_2' || \hat{\upsilon}' || \sigma_1 || M} \\
= \; & e(g, \hat{g})^{\alpha r} \\
= \; & e(\upsilon_1', \hat{\upsilon}')
\end{aligned}$$

and $e(\upsilon_2', \hat{g}) = e(g^{1/(st)}, \hat{g}) = e(g, \hat{\upsilon}')$.

Theorem 1: The simplified short group signature scheme is anonymous under the anonymity of Clarisse et al.'s short signature scheme.

Proof:

There are three roles in this game of the proof: an attacker $\mathcal{A}$ whose goal is the simplified short group signature scheme, a challenger $\mathcal{B}$ of the simplified short group signature scheme who also acts as an attacker to Clarisse et al.'s short signature scheme, and a challenger $\mathcal{C}$ of Clarisse et al.'s short signature scheme.

Setup: The challenger $\mathcal{C}$ generates the public parameters PP' of Clarisse et al.'s scheme and returns PP' to $\mathcal{B}$. Then, $\mathcal{B}$ selects $PP = (G_1, G_2, G_T, e, g, \hat{g}, X, \hat{X}, h)$ from PP' and sends it to $\mathcal{A}$ as the public parameters of the simplified scheme. Next, $\mathcal{C}$ generates the group master key gsk and group public key gpk by using the `GKeyGen` algorithm of Clarisse et al.'s scheme and sends gpk to $\mathcal{B}$. In addition, $\mathcal{B}$ constructs a group public key gpk' of the simplified scheme from gpk[2] and returns gpk' to $\mathcal{A}$.

Query:

- On the request to the $\mathcal{JO}$ oracle for the user i, $\mathcal{B}$ sends it to $\mathcal{C}$. $\mathcal{C}$ returns the result regarding whether the user i has already joined the group. Then, $\mathcal{B}$ sends the same result to $\mathcal{A}$.

[2]The group public key of Clarisse et al.'s scheme is $gpk = (\hat{A}_1 = \hat{g}^{\alpha_1}, \hat{A}_2 = \hat{g}^{\alpha_2}, \hat{B} = \hat{X}^{\alpha_2})$, while the group master key is (α_1, α_2). $\mathcal{B}$ chooses $(\hat{A}_2 = \hat{g}^{\alpha_2}, \hat{B} = \hat{X}^{\alpha_2})$ as the group public key of the simplified scheme, which implies the group master key is α_2.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2021.3062348, IEEE Transactions on Emerging Topics in Computing

- On the request to the $\mathcal{CO}$ oracle for the user i, $\mathcal{B}$ sends it to $\mathcal{C}$. $\mathcal{C}$ returns all the secret keys of the user i. Next, $\mathcal{B}$ sends the usk_i to $\mathcal{A}$.
- On the input of an index of user i and a message M to the $\mathcal{SO}$ oracle, $\mathcal{B}$ transmits the request to $\mathcal{C}$ with a random chosen $r \in Z_p$, and $\mathcal{C}$ returns $\sigma = (\upsilon_1', \upsilon_2', \hat{\upsilon}', \sigma_1, \sigma_2)$. Then, $\mathcal{B}$ keeps $(\upsilon_1', \upsilon_2', \hat{\upsilon}', \sigma_1)$ and constructs $\sigma_2' = X^{r/h(\upsilon_1'||\upsilon_2'||\hat{\upsilon}'||\sigma_1||M)}$. $\mathcal{B}$ returns $\sigma = (\upsilon_1', \upsilon_2', \hat{\upsilon}', \sigma_1, \sigma_2')$ as the response to $\mathcal{A}$.

Challenge: The attacker $\mathcal{A}$ sends the indices of two honest users i_1 and i_0 and a message M^* to $\mathcal{B}$. $\mathcal{B}$ sends them to the challenger $\mathcal{C}$ with a random $r^* \in Z_p$. $\mathcal{C}$ chooses a random bit b and computes a signature $\sigma^* = (\upsilon_1^{*'}, \upsilon_2^{*'}, \hat{\upsilon}^{*'}, \sigma_1^*, \sigma_2^*)$ on (i_b, M^*) by using the `Sign` algorithm of Clarisse et al.'s scheme. After receiving σ^*, $\mathcal{B}$ constructs $\sigma^{*'} = (\upsilon_1^{*'}, \upsilon_2^{*'}, \hat{\upsilon}^{*'}, \sigma_1^*, \sigma_2^{*'})$ where $\sigma_2^{*'} = X^{r^*/h(\upsilon_1^{*'}||\upsilon_2^{*'}||\hat{\upsilon}^{*'}||\sigma_1^*||M^*)}$, and returns it to $\mathcal{A}$.

Output: The attacker $\mathcal{A}$ outputs a guess b' for b to $\mathcal{B}$, and $\mathcal{B}$ transmits b' to $\mathcal{C}$ as his guess.

If $\mathcal{B}$ wins the game of Clarisse et al.'s scheme, then $\mathcal{A}$ wins the above game. Therefore, $ADV_{\mathcal{A}}^{Anon} \leq ADV_{\mathcal{B}}^{Anon}$. Since Clarisse et al.'s scheme is proved to have anonymity, $ADV_{\mathcal{B}}^{Anon}$ is negligible, and thus, $ADV_{\mathcal{A}}^{Anon}$ is also negligible. Therefore, our simplified scheme is also anonymous. ■

IV. Blockchain-based Edge Computing Data Storage Protocol

A. Edge Device in the Protocol

An edge device acts as a group manager in the group signature scheme, which also plays an important role in the blockchain-based edge computing data storage protocol. It not only manages the nearby end devices but also helps the data storage and transactions. The functions of an edge device in the protocol can be listed as follows.

- Manages the nearby end devices and sets up trust relationships with them. It generates a group signing key for each end device as a group manager.
- Creates transactions for the end devices in its domain. A valid transaction should involve the group signature signed by an end device. If the collected data are sensitive and require encryption, the edge device also helps the end devices to encrypt the data.
- Continuously collects data from the nearby end devices and forwards them to the storage cloud. It determines which storage address to use to store the data and sends the encrypted data to the address.

B. Blockchain in the Protocol

The blockchain acts as a platform to serve the edge computing data storage and protection. Unlike utilizing the blockchain as a cryptocurrency, a transaction in the edge computing data storage protocol is a request from an end device asking for services of data storage or data access. Regarding the discussion above, for protecting the privacy of end devices, the group signature can be utilized, and the edge device acts as the group manager [3]. For example, in smart health settings, which have a high privacy protection requirement, a patient's medical device A in a patients group with the group identity GID_P requests that it wants to store data in a certain medical cloud $Addr_{\mathcal{MC}}$. This transaction can be written as $T = (GID_P, Timestamp, Action = store\ \ data\ \ in\ \ Addr_{\mathcal{MC}})$. When a doctor's implantable medical device B in another doctor's group GID_D requests the data from A, it should post a transaction to the blockchain, which can be defined as $T = (GID_D, GID_P, Timestamp, Action = access\ \ data\ \ in\ \ Addr_{\mathcal{MC}})$. Therefore, the cloud with the address $Addr_{\mathcal{MC}}$ does not send the data to B until the transaction of the request has been verified and written in the blockchain.

C. Design of the Blockchain-based Edge Computing Data Storage Protocol

In the edge computing data storage protocol, we use our simplified group signature scheme for the anonymous authentication of end devices, since we assume that the edge device and the nearby end devices have set up long-term trust relationships and do not require the traceable property. The simplified group signature scheme can save more computational cost than the original group signature scheme and is especially suitable for the resource-limited IoT devices. To save space, we call the simplified group signature SGS, for short. To start, an IoT end device should register to the nearby edge device and obtain its group signing key. The edge device should also register to the blockchain for publishing the group public key. Next, an end device is able to post a transaction that can be verified by the blockchain. Our blockchain-based edge computing data storage protocol can be described as follows.

1) Registration Phase:

- An edge device acting as a group manager runs the `SGS.Setup` algorithm and broadcasts the public parameters PP. Assume that GID is the identity of the group. Following that, the edge device carries out the `SGS.GKeyGen` algorithm and obtains the group master/public key pair (gsk, gpk). Then, it appends (gpk, GID) to a request and sends it to the blockchain, where the blockchain miners are able to verify the group public key of the group GID.
- A nearby end device i that wants to join the group runs the interactive algorithm `SGS.Join` with the edge device and obtains its secret group signing key usk_i.

2) Transactions Generation and Verification Phase:

- After being successfully registered at the edge device, an end device i is able to send a request to store its data using the blockchain. Assume that the storage transaction is defined as $T_s = (GID, Timestamp, Action = store\ \ data\ \ in\ \ Addr_{\mathcal{C}})$, where $Addr_{\mathcal{C}}$ is the cloud address. The end device i signs T_s with its secret key usk_i by running the `SGS.Sign` algorithm. The edge device helps it to post this signed transaction to the blockchain.

[3] An edge device and the nearby end devices constitute a group.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2021.3062348, IEEE Transactions on Emerging Topics in Computing

- The blockchain miners verify the posted transaction using the corresponding group public key gpk. If the request transaction is verified, then it can be written into a block.

3) Data Validation Phase: The edge device helps the end device i to forward the whole data set to the storage cloud. Assume that the data set consists of n items and that $(h_1, \cdots, h_n)$ are the respective hash values for every item. Unlike the Merkel tree scheme with a logarithmic size of $O(log(n))$ proof, we design a scheme that only needs a proof size of $O(1)$.

- The end device i takes a security parameter λ as input, outputs a group $\mathcal{G}$ with prime order ρ, and δ is a generator over $\mathcal{G}$. Assume that $(h_1, \cdots, h_n)$ are the respective hash values for every item in its forwarded data set. The end device i keeps $A = \delta^{h_1 \cdots h_n}$ beforehand.
- If the end device i wants to validate whether the No.1 item of the data set has been correctly put into the cloud, it sends a request to the edge device. Then, the edge device sends $W = \delta^{h_2 \cdots h_n}$ and the No.1 item M_1, which is stored in the cloud to the end device i.
- The end device i first computes the hash value h_1' of M_1 and then validates it by the verification equation $A \stackrel{?}{=} W^{h_1'}$, which only requires 1 exponentiation operation.

4) Data Access Phase: An end device can set up an access control list that specifies clearly which group can access its data. To save space, the access control list is written as ACL for short.

- An end device i creates a transaction $T_a = (GID, Timestamp, ACL, Addr_{\mathcal{C}})$ with the group identity, access control list and the cloud address, generates a group signature using its group signing key usk_i appended to the transaction T_a, and publishes this transaction to the blockchain.
- The miners in the blockchain verify the appended signature using gpk, and if it is valid, then the transaction is written into the blockchain.
- If another IoT device in group GID' would like to access the stored data of i, then it can create a transaction $T_b = (GID', GID, Timestamp, Addr_{\mathcal{C}})$ and append the group signature.
- The miners in the blockchain should check the following two requirements:
  1) Whether the appended signature is valid.
  2) Whether GID' belongs to ACL. Only when T_b has passed the validation can it be written into the blockchain.
- The storage cloud checks whether the transaction requesting data exists in the blockchain. If it exists, then the cloud sends the data to the requester.

5) Data Trading Phase: Through the blockchain, any user can easily trade his or her data to some interesting parties. Assume that there is a buyer in the group $GID_{\mathbf{A}}$ and a seller in the group GID.

- An interested party (buyer) in the group $GID_{\mathbf{A}}$ posts a transaction $D_{\mathbf{A}} = (GID_{\mathbf{A}}, gpk_{\mathbf{A}}, GID, Deposit_{\mathbf{A}})$ appended with his group signature on $D_{\mathbf{A}}$. The transaction $D_{\mathbf{A}}$ includes the buyer's group identity, the buyer's group public key, the seller's group identity and a commitment of x dollars in $Deposit_{\mathbf{A}}$. The $Deposit_{\mathbf{A}}$ will be sent to the seller if he or she publishes a transaction and updates his or her access control list ACL by adding the $GID_{\mathbf{A}}$. If the seller does not sell the data after a given time t, the buyer can get his or her deposit back.
- If the seller would like to sell his data, then he or she can post a transaction $T_t = (GID, GID_{\mathbf{A}}, Timestamp, ACL, Addr_{\mathcal{C}})$ appended with his or her group signature with the help of the edge device. In the transaction T_t, the access control list has been updated and added $GID_{\mathbf{A}}$. Then, the seller posts another transaction $Getpaid$ appended with his or her group signature to request the x dollars in $Deposit_{\mathbf{A}}$.
- The miners in the blockchain verify T_t and $Getpaid$ to check whether $GID_{\mathbf{A}}$ has been added into ACL and whether $Getpaid$ has been created by the seller's group. If these two transactions are valid, then they can be written into the blockchain, and the buyer unlocks the x dollars in $Deposit_{\mathbf{A}}$.
- If the seller does not sell his or her data after the given time t, the buyer posts a transaction $Getdeposit$ to retrieve the deposited money.
- The miners in the blockchain verify $Getdeposit$ to check whether $Getpaid$ has been created by the seller's group and whether the t time has passed. If the transaction $Getdeposit$ is valid, then it can be written into the blockchain, and the buyer retrieve his or her money.

V. Security Analysis of the Protocol

A. Protocol Security

It is clear that the security of the proposed protocol is based on the security of the blockchain and SGS scheme. In each phase of the protocol, if the corresponding transactions can be verified by the blockchain, then they can be written into the blocks. The blockchain in this protocol works as a trusted third party, which is a very important role. The blockchain can verify the transaction of the data storing request and record the identity of the group and the storage address. In this way, it helps to manage data storage. When an end device posts a data request transaction, the blockchain can authenticate the requester. If the transaction is valid and written into the blockchain, then the cloud sends the data to the requester. In this way, the blockchain performs authentication instead of a trusted server. The security of the blockchain relies on the hardness of the sibyl attacks problem [16]. If an attacker can control the majority of the nodes in the blockchain, then it can make any arbitrary operations on the transactions. Thus, to make the blockchain secure enough, we should incentivize a sufficiently large number of miners in the blockchain. In our protocol, miners can acquire sufficient income from mining the blockchain from the following two aspects:

- Since the blockchain works as a trusted third party" instead of a trusted server, the service fees paid for a traditional server should be transferred to the miners in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2021.3062348, IEEE Transactions on Emerging Topics in Computing

the blockchain. The transaction fees can be deposited in advance, and once a transaction is completed, the miners can obtain the transaction fees immediately.

- The blockchain as a cryptocurrency creates block awards all the time, although we use it for the edge computing data services. Thus, the block awards can be given to the miners as their rewards.

Therefore, the miners will be attracted to the storage services for edge computing. The more edge computing applications utilize the blockchain, the more transaction fees are created, and the more miners are attracted to the blockchain. Thus, blockchain-based edge computing data storage will be secure.

Another main factor impacting the security of the protocol is the security of the SGS scheme, since verifying the transactions in the protocol mainly relies on the group signature. As discussed previously, our SGS scheme satisfies two security properties: *correctness* and *anonymity*. The property of *correctness* means that any user who has joined the group should be able to generate any valid group signature σ on any message M. On the other hand, an attacker outside the group cannot forge a valid signature, since it cannot obtain the group signing key usk_i. Thus, the blockchain platform based on our SGS scheme can be trusted.

B. Privacy

As discussed above, in some edge computing applications with higher privacy protection requirements, especially smart health, the data owner's identity should be preserved. Thus, when the data owner signs the transactions of a data storing request, an access control list or a data trading request, the owner does not hope to reveal his or her identity. For this problem, group signature is a good solution that keeps the signer's identity anonymous. In edge computing, the edge device naturally becomes a group manager to help other group members. Although we remove the two properties of *traceability* and *non-frameability* from our SGS scheme to save computational costs, our SGS scheme can still be proved anonymous, which means that the group member can sign on behalf of the group but that the group signature cannot be traced back to its issuer. Thus, the identity of the data owner can be preserved well in our protocol. In our scheme, we assume that the end devices and the nearby edge devices have setup a long-term trust relationships so that it is usually not required to trace the signer(end device)'s identity.

C. Resisting Denial of Services Attacks

Any devices accessing the data in a cloud address will be recorded in the blockchain, and there is no way to deny this operation. The data owner can know which groups have accessed his or her data, and unauthorized groups that attempt to access data will be detected and blocked by the blockchain, which can largely prevent denial of services attacks. In our protocol, we assume that the edge devices (group managers) are convinced, since the cloud would issue credentials to the edge devices before the end devices start trusting to them. If an authorized edge device is compromised, the cloud should revoke its credential and notify it to the nearby end devices.

TABLE I: Performance and Security Model Comparison

Performance and Security	Our SGS scheme	Clarisse's scheme	Boneh's scheme
Sign	$5Ex + 4Mul$	$6Ex + 5Mul$	$3Pr + 12Ex + 5Mul$
Verify	$5Pr + 1Ex + 1Mul$	$5Pr + 1Ex + 1Mul$	$5Pr + 12Ex + 8Mul$
Signature Size	1536 bits	1536 bits	2304 bits
Security	Standard Model	Standard Model	Random Oracle

D. Data Storage Validation

In our protocol, the edge device helps the end devices to forward the whole data set to the storage cloud. If the data owner wants to inquire whether an important item in his or her data set has been well stored, then the edge device should provide a proof. Unlike the logarithmic-size proof of the Merkel tree scheme, we design an efficient data validation scheme that has a proof size of only $O(1)$, but it is secure enough. The root hash value $A = \delta^{h_1 \cdots h_n}$ of the data set is kept by the data owner in advance, and the edge device sends the data owner the inquired item M_1 and the product of the hash values for other items $W = \delta^{h_2 \cdots h_n}$. Then, the data owner can compute the hash value h_1' of M_1 and compare $W^{h_1'}$ with A to validate the item M_1. If $A \neq W^{h_1'}$, then M_1 is detected to be modified. Obviously, the security of the proposed validation scheme relies on the hardness of the discrete logarithmic problem and the collision resistant hash function. Thus, an attacker cannot forge a proof for a modified M_1.

VI. Performance Analysis

The performance of our protocol mainly relies on our SGS scheme, and thus we should firstly compare our SGS scheme with other group signatures used in the proposals combining blockchain and edge computing techniques. As we know, there are only two such group signature schemes. One is Boneh et al.'s short group signature [28] used in [29], and the other is Zhang et al.'s modified group signature [30] used in their scheme. However, the most important property - anonymity of group signature is eliminated from Zhang et al.'s scheme, and so it cannot be considered as a group signature. Therefore, we only compare our SGS scheme with Boneh et al.'s short group signature scheme [28] and Clarisse et al.'s scheme [34]. Assume that the bilinear groups are generated by using B-N curves [34] and that the representation length of the elements in Z_p and G_1 is 256 bits, while the representation length of the elements in G_2 is 512 bits. Let Pr denote the bilinear pairing operation, Ex denote the exponentiating operation, and Mul denote the multiplicating operation. Table I shows the performance and security model comparison of group signature schemes. For generating a signature, the computational cost of our SGS scheme is less than the other two schemes, especially Boneh et al.'s short group signature scheme, since the traceability has been removed.

To test the computational cost of our protocol further, we implement it over two different platforms. The first platform

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2021.3062348, IEEE Transactions on Emerging Topics in Computing

TABLE II: Time Cost of End Device in the Registration Phase

Platform	MacBook	Edison
End Device	0.0157 s	0.31 s

TABLE III: Time Cost of End Device in the Transaction Generation and Verification Phase

Platform	MacBook	Edison
End Device	0.0254 s	0.5 s

is a MacBook Pro with an Intel core i5 CPU (2.7 GHz) running Os X 10.11.6, with 8 GB RAM. The second is an Intel Edison development platform with a dual-core, dual-threaded Intel Atom CPU, at 500 MHz with 1 GB RAM, running Yocto Linux v1.6. In IoT systems, some of the edge devices are also relatively resource-constrained, so the Intel Edison development platform is considered a good choice to rapidly prototype the edge devices in edge computing. We implement the algorithms in C by using the pairing-based cryptography (PBC) library[38], which has been implemented in the basic arithmetic and pairing operations. There are seven types of curves in PBC, and we choose the fastest Type-A curves for the implementation.

In the registration phase of the protocol, the end device only requires 2 exponentiations, while the edge device needs 3 exponentiations for each new joined end device. Table II shows the time costs of each end device over two platforms. Fig. 3 shows time costs of the edge device over two platforms. In the transactions generation and verification phase, only the end device needs to carry out the `SGS.Sign` algorithm, and it requires 5 exponentiations and 4 multiplications. The edge device does not need to execute any cryptographic operations. Table III shows the time costs of the end device in the transactions generation and verification phase. It only requires 0.5 s over the resource-constrained Edison platform, which means that our SGS scheme is efficient enough for the practical edge computing applications. In the data validation phase, the edge device requires the execution of 1 exponentiation and n multiplications, while the end device only needs 1 exponentiation. Table IV shows that the time cost of the end device in the data validation phase is very small, which also explains the efficiency of our data validation scheme with a proof size of $O(1)$. In the data access phase, regardless of whether it is the data owner's end device or the requester's end device, the `SGS.Sign` algorithm is only required to be carried out one time. Thus, the time cost of the end device in this phase is the same as the time cost of the end device in the transactions generation and verification phase. In the data trading phase, regardless of whether it is the buyer's end device or the seller's end device, the `SGS.Sign` algorithm is required to be executed two times. Table V shows the time cost of the end device in the data trading phase, which is also very fast.

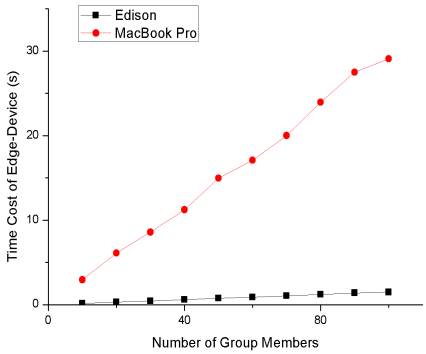

Fig. 3: Time Cost of Edge Device in the Registration Phase

TABLE IV: Time Cost of End Device in Data Validation Phase

Platform	MacBook	Edison
End Device	0.008 s	0.152 s

From the above experiments, it can be concluded that our SGS scheme is efficient enough for the practical applications in the proposed protocol, and the computational load of our data validation scheme is also very lightweight.

VII. Conclusion

To design an efficient and privacy-protecting blockchain-based edge computing data storage protocol, we first construct a simplified group signature scheme from which two secure properties have been removed to improve the efficiency. Through the cloud services, the edge device can easily set up long-term trust relationships with the nearby end device, and thus, our SGS scheme does not need the traceable and non-frameable properties. Based on the SGS scheme, we design a blockchain-based edge computing data storage protocol, where the blockchain works as the trusted third party to ensure the security of the proposed protocol. Moreover, the privacy of the user's identity can also be preserved by the SGS scheme. Another contribution in our protocol is that we design an efficient data validation scheme that has a proof size of only $O(1)$. We test our protocol over the Intel Edison platform and find that it performs well enough for practical applications.

ACKNOWLEDGMENTS

Thanks Dr. J.T.H. Yuen for discussing this paper! This research is partially supported by the National Natural Science

TABLE V: Time Cost of End Device in the Data Trading Phase

Platform	MacBook	Edison
End Device	0.048 s	0.93 s

Foundation of China under Grant No.61672016, the Qing Lan Project of Jiangsu Province and the Six talent peaks project in Jiangsu Province under Grant No. RJFW010.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future Generation Computer Systems, vol. 29, no. 7, pp. 1645-1660, 2013.

[2] R. Li, T. Song, N. Capurso, J. Yu, J. Couture, and X. Cheng, Iot applications on secure smart shopping system, IEEE Internet of Things Journal, vol. 4, no. 6, pp. 1945C1954, 2017.

[3] T. Song, R. Li, B. Mei, J. Yu, X. Xing, and X. Cheng, A privacy preserving communication protocol for iot applications in smart homes, IEEE Internet of Things Journal, 2017.

[4] 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. ACM, 2012, pp. 13-16.

[5] I. Stojmenovic and S. Wen, The fog computing paradigm: Scenarios and security issues, in Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on. IEEE, 2014, pp. 1-8.

[6] K. Sha, W. Wei, A. Yang, and W. Shi, Security in internet of things: Opportunities and challenges, in Proceedings of IIKI 2016, Oct 2016.

[7] Ranadheer Errabelly, Kewei Sha, Wei Wei, T. Andrew Yang, and Zhiwei Wang. EdgeSec: Design of an Edge Layer Security Service to Enhance IoT Security, IEEE International Conference on Fog and Edge Computing (ICFEC 2017).

[8] Kewei Sha, Wei Wei, T. Andrew Yang, Zhiwei Wang, Weisong Shi:On security challenges and open issues in Internet of Things. Future Generation Comp. Syst. 83: 326-337 (2018)

[9] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains" Ph.D. dissertation, University of Guelph, 2016.

[10] M. Vukoli'c, The quest for scalable blockchain fabric: Proof-of-work vs. bft replication, in International Workshop on Open Problems in Network Security. Springer, 2015, pp. 112-125.011.

[11] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin: A scalable blockchain protocol." in NSDI, 2016, pp. 45-59.

[12] Laizhong Cui, Shu Yang, Ziteng Chen, Yi Pan, Zhong Ming, and Mingwei Xu. A Decentralized and Trusted Edge Computing Platform for Internet of Things, IEEE Internet of Things Journal, 7(5): 3910-3922, 2020.

[13] I-Hsun Chuang, Shih-Hao Huang, Wei-Chu Chao, Jen-Sheng Tsai, and Yau-Hwang Kuo. TIDES: A Trust-Aware IoT Data Economic System With Blockchain-Enabled Multi-Access Edge Computing, IEEE Access, 8: 85840-85855, 2020.

[14] Shaoyong Guo, Xing Hu, Song Guo, Xuesong Qiu, and Feng Qi. Blockchain Meets Edge Computing: A Distributed and Trusted Authentication System, IEEE Transactions on Industrial Informatics, 16(3):1972-1983, 2020.

[15] Ma Zhaofeng, Wang Xiaochang, Deepak Kumar Jain, Haneef Khan, Gao Hongmin, and Wang Zhen, A Blockchain-Based Trusted Data Management Scheme in Edge Computing, IEEE Transactions on Industrial Informatics, 16(3):2013-2021, 2020.

[16] Ruinian Li, Tianyi Song, Bo Mei, Hong Li, Xiuzhen Cheng, Liming Sun. Blockchain For Large-Scale Internet of Things Data Storage and Protection. IEEE Transactions on Services Computing, 12(5): 762-771, 2019.

[17] Jing Wang, Libing Wu, Kim-Kwang Raymond Choo, and Debiao He. Blockchain-Based Anonymous Authentication With Key Management for Smart Grid Edge Computing Infrastructure, IEEE Transactions on Industrial Informatics, 16(3):1984-1992, 2020.

[18] Bonnah Ernest, Ju Shiguang. Privacy Enhancement Scheme (PES) in a Blockchain-Edge Computing Environment, IEEE Access, 8: 25863-25876, 2020.

[19] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In ASIACRYPT, pages 552-565. Springer-Verlag, 2001.

[20] David Chaum and Eugene van Heyst. Group signatures. In Donald W. Davies, editor, EUROCRYPT'91, volume 547 of LNCS, pages 257-265. Springer, Heidelberg, April 1991.

[21] X. Huang, W. Susilo, Y. Mu, and F. Zhang, On the security of certificateless signature schemes from asiacrypt 2003, in CANS, vol. 2005, no. 3810. Springer, 2005, pp. 13-25.

[22] S. S. Chow, C. Boyd, and J. M. G. Nieto, Security-mediated certificateless cryptography, in Public key cryptography, vol. 3958. Springer, 2006, pp. 508-524.

[23] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, CRYPTO 2000, volume 1880 of LNCS, pages 255-270. Springer, Heidelberg, August 2000.

[24] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, EUROCRYPT 2003, volume 2656 of LNCS, pages 614-629. Springer, Heidelberg, May 2003.

[25] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, CT-RSA 2005, volume 3376 of LNCS, pages 136-153. Springer, Heidelberg, February 2005.

[26] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, SCN 10, volume 6280 of LNCS, pages 381-398. Springer, Heidelberg, September 2010.

[27] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, CT-RSA 2016, volume 9610 of LNCS, pages 111-126. Springer, Heidelberg, February / March 2016.

[28] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures", in Advances in CryptologyłCRYPTO, M. Franklin, Ed., Santa Barbara, CA, USA: Springer, 2004, pp. 41-55.

[29] Keke Gai, Yulu Wu, Liehuang Zhu, Lei Xu, and Yan Zhang. Permissioned Blockchain and Edge Computing Empowered Privacy-Preserving Smart Grid Networks, IEEE Internet of Things Journal, 6(5):7992-8004, 2019.

[30] Shijie Zhang and Jong-Hyouk Lee. A Group Signature and Authentication Scheme for Blockchain-Based Mobile-Edge Computing, IEEE Internet of Things Journal, 7(5):4557-4565, 2020.

[31] David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier Lopez, and Taesoo Kim, editors, ASIACCS 18, pages 551-565. ACM Press, April 2018.

[32] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, ASIACRYPT 2014, Part I, volume 8873 of LNCS, pages 491-511. Springer, Heidelberg, December 2014.

[33] Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Signatures with exible public key: A unified approach to privacy-preserving signatures (full version). IACR Cryptology ePrint Archive, 2018:191, 2018.

[34] Remi Clarisse and Olivier Sanders. Short Group Signature in the Standard Model. IACR Cryptology ePrint Archive, 2018:1115, 2018.

[35] Hamed Tohidi, Vahid Tabataba Vakili. Lightweight authentication scheme for smart grid using Merkle hash tree and lossless compression hybrid method, IET Communications, 12(19): 2478 - 2484, 2018

[36] Melesio Calderon Munoz, Melody Moh, Teng-Sheng Moh. Improving smart grid authentication using Merkle Trees, 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), pages. 793 - 798, 2014

[37] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Euf-cma-secure structure-preserving signatures on equivalence classes. IACR Cryptology ePrint Archive, 2014:944, 2014.

[38] B. Lynn., The pairing-based cryptography (pbc) library, http://crypto.stanford.edu/pbc.

Zhiwei Wang was born in Yangzhou, Jiangsu,China, in 1976. He received the Ph.D. degree in Cryptography from the Beijing University of Posts and Telecommunications in 2009. He is currently a professor at the school of computer, Nanjing University of Posts and Telecommunications. He was a research associate at the University of Hong Kong from March 2014 to March 2015. He has published more than 50 journal articles and referred conference papers. His research interests include applied cryptography, security and privacy in mobile and wireless systems, clouding computing, and fog/edge computing.