Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5565663, 12 pages
https://doi.org/10.1155/2021/5565663

Research Article

Decentralized Data Aggregation: A New Secure Framework Based on Lightweight Cryptographic Algorithms

Xin Xie and Yu-Chi Chen

Department of Computer Science and Engineering, Yuan Ze University, Taoyuan 324, Taiwan

Correspondence should be addressed to Yu-Chi Chen; wycchen@ieee.org

Received 8 January 2021; Revised 25 February 2021; Accepted 23 March 2021; Published 15 April 2021

Academic Editor: SK Hafizul Islam

Blockchain has become very popular and suitable to the Internet of Things (IoT) field due to its nontamperability and decentralization properties. The number of IoT devices and leaders (who own IoT devices) is increased exponentially, and thus, data privacy and security are undoubtedly significant concerns. In this paper, we summarize some issues for the BeeKeeper system, a blockchain-based IoT system, proposed by Zhou et al., and then aim for presenting an improved solution for decentralized data aggregation (DDA) on IoT. Firstly, we formally state the security requirements of DDA. Secondly, we propose our basic DDA system by using secret sharing to improve its efficiency and smart contracts as the computing processors. Moreover, the proposed full-fledged system achieves data sharing (e.g., a leader to access data of others' devices), which is realized by using local differential privacy and cryptographic primitives such as token-based encryption. Finally, to show the feasibility, we provide some implementations and experiments for the DDA systems.

1. Introduction

The Internet of Things (IoT) is the extensive concept of Internet with physical devices for jointly realizing a service or achieving target functionality. For example, these devices (embedded systems or sensors) communicate and interact with others over the Internet and also can be remotely monitored and controlled. In fact, IoT has offered numerous applications in the aspects of consumer, commerce, industry, and infrastructure. For education purposes, Raspberry Pi plays the role of a cheap and fundamental platform to develop applications of IoT. Typically, in IoT systems, the purpose for collecting data may be used in the future (e.g., analysis for behaviors of consumers), and such data are stored in the centralized cloud storage. Hence, users are enforced to trust that the centralized cloud will protect their unencrypted data.

This suffices to wrap up the following issues for centralized cloud-based IoT systems.

(i) Centralization: the cloud is in charge of providing and maintaining the service. If it crashes, the centralized system is no longer to work [1]. For example, denial of service attacks may focus on the single cloud to terminate the functionality

(ii) Trust: the users outsource their data to the cloud [2] and only hold limited control for their data. They have to fully trust the cloud which does not modify and delete the stored data

(iii) Scalability: more efficient IoT devices and larger bandwidth communication produce big data streams [3]. Therefore, the centralized cloud must be accordingly efficient or scalable for tackling those data

(iv) Data security: the stored data are unencrypted. Sensitive data are directly revealed to the cloud. However, encryption may be a solution, but processing

over encrypted data is always an issue. In the past, plenty of clever notions [4–6] are proposed to address this

Blockchain, introduced by Nakamoto [7], is referred to as the core technology of Bitcoin. It can be regarded as the distributed ledger consisting of blocks chained in order. The potential advantages of blockchain for improving IoT systems are as follows:

(i) Decentralization and tamper proof [8]: the property of decentralization in blockchain frees our IoT system from setting up centralized nodes. Anyone (a.k.a. a node or participant in the protocol) can join or leave the protocol execution at any time without getting permission from an authority. In general, it is an effective method for maintaining a public, immutable, and ordered ledger of records (for instance, in the Bitcoin, these records are simply transactions); that is, records can be added to the end of the ledger at any time (but only to the end of it); additionally, it is guaranteed that records previously added cannot be removed or reordered and that all honest nodes have a consistent view of the ledger, which is referred to as consistency

(ii) Anonymous: since the data exchange between the nodes of the blockchain follows a fixed and predictable algorithm, the blockchain network is trustless and can exchange data based on addresses instead of identities

(iii) Smart contracts (SC): SC are a powerful application based on blockchain, where SC programs deployed on the blockchain and can be triggered or executed by events. SC can also make our IoT devices smarter since the behavior of them can be specified by SC

In a sense, blockchain seems a sort of potential solution to the above-mentioned issues in the IoT scenario, including centralization, trust, and scalability, since it provides decentralized services, distributed trust, and permissionless property for scalability.

1.1. Related Work. Very recently, Zhou et al. [9] proposed a blockchain-based threshold IoT system, named as BeeKeeper. This system applies secret sharing to encrypt data. In this system, there are three entities: a leader, the leader's devices, and a certain number of servers, and all of them only communicate with blockchain. The threshold is denoted by t -out-of-n ($n > t$), which implies that there are totally n servers, and any t suffice to complete the task. Let us keep highlighting the procedure of the system. Devices collect data and then submit them to blockchain. Servers will retrieve them from blockchain, run some operations, and return the processed data back to blockchain. Finally, the leader can submit his/her request to blockchain and obtain the (aggregation) result as long as at least t servers complete the processing over the data. In fact, the same authors [10] have proposed an improvement named as BeeKeeper 2.0, which can offer some additional functionalities, but the overall framework is not significantly changed. In addition to data aggregation, blockchain can be also applied to key management in IoT [11].

1.2. Contributions. The motivation of this paper is the observation in BeeKeeper, regarding a few concerns briefly shown as follows. (1) At least t servers must be maintained to operate for keeping functionality. (2) Blockchain seems only the communication platform and database. (3) The underlying secret sharing is realized by using polynomials. Our main results are conceptually simple solutions where we use smart contracts (supporting data aggregation) to overcome (1) and (2) and then rely on secret sharing to improve efficiency to (3). We propose a warm-up system which is composed of the above building blocks and achieves decentralized data aggregation on IoT. However, in the warm-up, the devices are of the leader, which means that he/she cannot obtain the results collected by the other leaders' devices. It is an extensional feasibility, but indeed induces privacy issues for accessing others' data. Hence, we provide a full-fledged system to achieve this by using local differential privacy and token-based encryption. Accordingly, we also state the security definition for our systems, and under such the definition prove their security. As shown in Table 1, we briefly explain the difference from the previous works. The proposed DDA/DDA$^+$ does not need to set up a server to handle complex calculations. The aggregation of data is left to record nodes of DDA/DDA$^+$ or the owner of the smart contract (i.e., leader), which operates with the (simple) logic design. For the verification function, Zhou et al. [9, 10] require additional verification for all participants. However, in our DDA$^+$ scheme, we only need to consider devices to verify the legitimacy of the leader, for which we provide a token tk. In the high level DDA$^+$, data privacy protection is added. We use local differential privacy to protect sensitive data of devices over gathering. In addition, our smart contract sets up multiple functions, not just processing transfer operations. Finally, we complete the security proof of the program.

1.2.1. Comparisons to the Previous Version [12]. In the full version, we go through the main ideas of protocol design and then give a solid description of security. In addition, we present the full-fledged construction from local differential privacy and prove the security in a formal way. Finally, experiments are provided to show its effectiveness.

1.3. Organization. The remainder of this paper is organized as follows. First of all, Section 2 briefly describes the basic technologies of the decentralized data aggregation (DDA) system. Then, in Section 3, we present the system model and security requirements of this system. Thereafter, the warm-up and full-fledged DDA systems are described in Sections 4 and 5, respectively. Implementations and analysis of the systems are given in Section 6. Finally, we provide the conclusions of this paper in Section 7.

Table 1: Comparison with previous works.

BeeKeeper [10] BeeKeeper$^+$ [9]		DDA	DDA$^+$
Leader	✓	✓	✓
Devices	✓	✓	✓
Servers	✓	×	×
Verification	✓	×	✓
Privacy protecting	×	×	✓
Data aggregation	Servers	Nodes/leader	Nodes/leader
Smart contract	Transfer	Multifunction	Multifunction
Security	No proof	SDS, NDS, SCS, OS	SDS, NDS, SCS, OS

2. Preliminaries

In this section, we briefly introduce some preliminaries. We use λ to denote security parameters and PPT to denote polynomial time adversaries. Also, the assumptions are generated by the λ parameter; we define a negligible function for any polynomial function $\text{poly}(\lambda)$ and satisfies λ sufficiently large, $\text{negl}(\lambda) \le (1/\text{poly}(\lambda))$ holds.

2.1. Secret Sharing. Secret sharing [13, 14] is a notion that divides a secret into shares. The secret can be recovered by combining certain numbers of shares. Here, secret sharing for n participants only allows n shares together to recover the secret, so-called (n, n) secret sharing. Also, for some specific functionality, we state the definition as follows.

Additive secret sharing: a secret message secret will be split into n shares (share_1, share_2,…, share_n) in (n, n) secret sharing scheme. This scheme consists of five phases.

(i) ShareGen (1^λ, n, secret): given security parameter 1^λ, device numbers n, secret $\in \mathbb{S}$ as input, then outputs share_1,…, $\text{share}_n \in \{0,1\}^\ell$

(ii) SecretRec (share_1,…, share_n): taking all the shares share_1,…, share_n as input, this algorithm can output secret

(iii) AddOnShare (share, x): it takes data x, share as input, and then generates share^x

(iv) PartialRec ($\text{share}_1^{x_1}$,…, $\text{share}_n^{x_n}$): this algorithm takes $\text{share}_1^{x_1}$,…, $\text{share}_n^{x_n}$ as input and then outputs partrec as the result for partial recovery

(v) Extract (partrec, secret): it takes partrec and secret as input and returns sumx

We give the formal security definition of (n, n) secret sharing as follows.

Definition 1. Let n be the number of participants, which implies that only collecting n shares can recover the secret. Accordingly, any $n-1$ shares cannot get any information about the secret. The probability of an (unbounded) adversary $\mathcal{A}$ is

$$\Pr\left[\mathcal{A}\left(\text{share}_{j_1}, \cdots, \text{share}_{j_{n-1}}\right) \to \text{secret}\right] = \frac{1}{|\mathbb{S}|}, \quad (1)$$

where $j_1, \cdots, j_n - 1 \in \{x \mid 1 \le x \le n, x \in N\}$ and $j_1, \cdots, j_{n-1}$ is not repeated.

The construction can be easily realized by using additive secret sharing.

2.2. Smart Contract-Based Aggregation Service. The smart contract-based aggregation service can mainly achieve identification of devices and data aggregation. Note that smart contracts do not provide any confidentiality. In general, a smart contract-based aggregation service is composed of the following algorithms (The SC provides the specified functionality for the specified authorized user and provides data signature and verification, since the underlying blockchain works with signatures).

(i) InitContract(ϱ): given only the security parameters ϱ as input, it initially creates and deploys the personal smart contract on the blockchain and finally outputs the corresponding smart contract address (*SC.Address*)

(ii) AddDevice(*SC.Address,leaderId,id,* $\mathcal{Q}$): it takes the smart contract address *SC.Address*, the personal identity of the system leader *leaderId*, the devices' id *id*, and the id list $\mathcal{Q}$ (all $id \in \mathcal{Q}$), and then, the algorithm generates as inputs and outputs $flag_{id}$ (*flag* indicates that the device holds permissions to execute certain functions in SC) of every id and updated id list $\mathcal{Q}'$

(iii) Upload(*SC.Address,id,flag,m,*$\mathcal{DB}$): it takes the smart contract address *SC.Address*, the devices' id *id*, the corresponding flag *flag*, the collected data *m*, and the database $\mathcal{DB}$ (all $m \in \mathcal{DB}$) as inputs. Subsequently, the algorithm can generate and output the result database $\mathcal{DB}'$

(iv) Aggregation(*SC.Address,* $\mathcal{DB}'$ *,leaderId*): it takes the smart contract address *SC.Address*, the result database DB^0, and the personal id of system leader *leaderId* as inputs. Finally, it returns the aggregation result x

The smart contract-based aggregation service system holds the following security properties.

Definition 2. The probability that a polynomial adversary $\mathcal{A}$ without *leaderId* and corresponding conditions in this system can execute the *AddDevice* algorithm is negligible, that is,

$$\Pr\left[\text{AddDevice}_{\mathcal{A}}(\text{SC.Address},\perp,\text{id},\mathcal{Q}) \longrightarrow \text{flag}_{\text{id}}, \mathcal{DB}'\right] \leq \text{negl}(\lambda), \tag{2}$$

where id denotes the device $(\text{id} \in \mathcal{Q})$.

Definition 3. The probability that the polynomial adversary $\mathcal{A}$ as a device without corresponding flag in this smart contract-based aggregation service system can upload data is negligible, that is,

$$\Pr\left[\text{Upload}_{\mathcal{A}}(\text{SC.Address}, \text{id},\perp,m, \mathcal{DB}) \longrightarrow \mathcal{DB}'\right] \leq \text{negl}(\lambda), \tag{3}$$

where m denotes the data uploaded $(m \in \mathcal{DB})$.

Definition 4. The probability that a polynomial adversary $\mathcal{A}$ without leaderId and corresponding conditions in this system can execute the *Aggregation* algorithm is negligible, that is,

$$\Pr\left[\text{Aggregation}_{\mathcal{A}}\left(\text{SC.Address}, \mathcal{DB}',\perp\right) \longrightarrow x\right] \leq \text{negl}(\lambda). \tag{4}$$

3. System Model

3.1. Syntax of Decentralized Data Aggregation. In this section, we aim for a concrete notion, called decentralized data aggregation (DDA). We formally present the system model below. In a nutshell, DDA consists of three phases, denoted by DDA = (Initialization, DataCollection, and DataReconstruction). Some notations are listed in Table 2. The DDA syntax is formally described as follows.

(i) Initialization $(1^\lambda, \varrho, n, L.\text{EOA}, D_i.\text{EOA}, \mathcal{S}, \mathcal{Q})$: the algorithm takes as inputs the security parameter 1_λ, ϱ, device quantity n, leader's public address L.EOA, public address of the i-th device D_i.EOA, leader's private key $\mathcal{S}$, device address set $\mathcal{Q}$ and outputs $\mathcal{K}_i$, *SC.Address*, $flag_{L_{id}}$, $\mathcal{Q}'$

(ii) Data Collection (*SC.Address*, L.EOA, m_i, $\mathcal{DB}$, $\mathcal{K}_i$): the algorithm takes as inputs contract address *SC.Address*, leader's public address L.EOA, the raw data for the i-th device m_i, device i stored database DB, the private key of the i-th device K_i and outputs C

(iii) Data Reconstruction (C, S): the algorithm takes as inputs ciphertext C, leader's private key S, and outputs M

Table 2: Notations.

Symbol	Description
$1^\lambda, \varrho$	The security parameters[1]
n	Number of devices planned to be deployed
D_i	The device D_i belongs to the leader L
D_i'	The device D_i' does not belong to the leader L
$\mathcal{Q}/\mathcal{NQ}$	The collection of device IDs
$\mathcal{DB}/\mathcal{DB}'$	The identity of D_i or D_i's database
X.EOA	The public address of the entity X running on platform
X.EPK	The private address of the entity X running on platform
SC.Address	The smart contract address
$\mathcal{S}/\mathcal{S}'$	A secret value set by the leader
$\mathcal{K}_i/K_i'$	The shared key value
$\mathbb{S}$	The domain of secret key
rnd	The number of rounds of device update data

[1]These two different security parameters are used to initialize the different functions.

3.2. Security Requirements. In our security model, privacy-preserving aggregation is our main concern. Since the smart contract is completely public, we assume that there are no inside threats in which the contract code is secure. To clarify our security definition, we assume that there are only three parties on the system: the leader, the smart contract, and the device belonging to the leader's IoT. We elaborate the following security definitions for all probability polynomial time adversaries A where we can quantify its computation is polynomially bounded such as $\text{poly}(\lambda)$.

(1) *Single Device Security (SDS).* We assume an adversary $\mathcal{A}$ who knows the probability distribution of the plaintext space M of one of the devices, but does not have any knowledge of the key. $\mathcal{A}$ can eavesdrop on communication (leader and IoT device) and thus observe the ciphertext of an IoT device. Let us briefly describe this simple security experiment $\text{Exp}^{\text{SDS}}_{\text{DDA},\mathcal{A}}$ as follows:

(i) Setup: the challenger randomly selects $\mathcal{S}$ and obtains K_i sequence $(i = 1, \cdots, n)$, and then, $\mathcal{A}$ outputs a pair of message $m_0, m_1 \in M$

(ii) Challenge phase: the challenger randomly selects K_i and a bit $b \in \{0, 1\}$, and then, $C \leftarrow$ DataCollection $(\text{SC.Address}, L.\text{EOA}, m_b,\perp,K_i)$ is computed and

given to $\mathscr{A}$. We refer to C as the challenge ciphertext. Note that the ciphertext C only has a ciphertext generated by a single device operation

(iii) Guess: $\mathscr{A}$ outputs a bit b'

We defined $A^{`}$s advantage as

$$\mathrm{Adv}^{\mathrm{SDS}}_{\mathrm{DDA},\mathscr{A}} = \left|\Pr\left[\mathrm{Exp}^{\mathrm{SDS}}_{\mathrm{DDA},\mathscr{A}}(\lambda) = 1\right]\right|. \quad (5)$$

(2) $n-1$ *Threshold Device Security (NDS)*. We assume an adversary $\mathscr{A}$ who is given $n-1\mathscr{K}$ values, but cannot infer $\mathscr{S}$. Collecting the knowledge of $n-1$ pieces reveals no significant information about $\mathscr{S}$. Let us briefly describe this simple security experiment $\mathrm{Exp}^{\mathrm{NDS}}_{\mathrm{DDA},\mathscr{A}}$ as follows:

(i) Setup: the challenger randomly selects S and obtains K_i sequence $(i = 1, \cdots, n)$

(ii) Challenge phase: the challenger randomly select $n-1\mathscr{K}$ values to send to $\mathscr{A}$

(iii) Guess: $\mathscr{A}$ outputs $\mathscr{S}'$

Definition 5 (SDS). We say that DDA meets Single Device Security if for all probabilistic polynomial time adversary $\mathscr{A}$, $\mathrm{Adv}^{\mathrm{SDS}}_{\mathrm{DDA},\mathscr{A}} \le (1/2) + \mathrm{negl}(\lambda)$.

We defined A's advantage as

$$\mathrm{Adv}^{\mathrm{NDS}}_{\mathrm{DDA},\mathscr{A}} = \left|\Pr\left[\mathrm{Exp}^{\mathrm{NDS}}_{\mathrm{DDA},\mathscr{A}}(\lambda) = 1\right]\right|. \quad (6)$$

(3) *Smart Contract Security (SCS)*. We assume an adversary A (A.EOA) who tries to modify the uploaded data. Let us briefly describe this simple security experiment $\mathrm{Exp}^{\mathrm{SCS}}_{\mathrm{DDA},\mathscr{A}}$ as follows.

Definition 6 (NDS). We say that DDA meets $n-1$ Threshold Device Security if for all probabilistic polynomial time adversary $\mathscr{A}$, $\mathrm{Adv}^{\mathrm{NDS}}_{\mathrm{DDA},\mathscr{A}} \le \mathrm{negl}(\lambda)$.

L.EOA must authorize A.EOA to enter SC, that is, run AddDevice, $\mathscr{A}$ does not get L.EOA's flag $flag_{id}$.

(i) Setup: the challenger deploys the smart contract and returns *SC.Address* to $\mathscr{A}$

(ii) Challenge phase: the challenger randomly selects q identities of devices to run AddDevice

(iii) Guess: $\mathscr{A}$ outputs $flag_{id}$. We defined $\mathscr{A}^{`}$s advantage as

$$\mathrm{Adv}^{\mathrm{SCS}}_{\mathrm{DDA},\mathscr{A}} = \left|\Pr\left[\mathrm{Exp}^{\mathrm{SCS}}_{\mathrm{DDA},\mathscr{A}}(\lambda) = 1\right]\right|. \quad (7)$$

(4) *Outlier Security (OS)*. We assume that the external adversary A gets the result of all ciphertext aggregation. Let us briefly describe this simple security experiment $\mathrm{Exp}^{\mathrm{OS}}_{\mathrm{DDA},\mathscr{A}}$ as follows:

(i) Setup: the challenger randomly selects $\mathscr{S}$, and the set of $\mathscr{K}$ is generated by the Initialization algorithm $(i = 1, 2, \cdots, n)$. $\mathscr{A}$ outputs a pair of message $m_0, m_1 \in M$

(ii) Challenge phase: the challenger randomly chooses a bit $b \in \{0, 1\}$ and generates ciphertext aggregated by n devices by running $C \leftarrow \mathrm{DataCollection}(\mathrm{SC.Address}, L.\mathrm{EOA}, m_b, \mathscr{DB}, \mathscr{K}_i)$ and gives C to $\mathscr{A}$

(iii) Guess: $\mathscr{A}$ outputs a bit b'

Definition 7 (SCS). We say that DDA meets Smart Contract Security if for all probabilistic polynomial time adversary $\mathscr{A}$,

$$\mathrm{Adv}^{\mathrm{SCS}}_{\mathrm{DDA},\mathscr{A}} \le \mathrm{negl}(\lambda). \quad (8)$$

We defined A's advantage as$\mathrm{Adv}^{\mathrm{OS}}_{\mathrm{DDA},\mathscr{A}} = |\Pr[\mathrm{Exp}^{\mathrm{OS}}_{\mathrm{DDA},A}(\lambda) = 1]|$.

Definition 8 (OS). We say that DDA meets Outlier Security if for all probabilistic polynomial time adversary $\mathscr{A}$,

$$\mathrm{Adv}^{\mathrm{OS}}_{\mathrm{DDA},\mathscr{A}} \le \frac{1}{2} + \mathrm{negl}(\lambda). \quad (9)$$

4. Warm-Up: A Basic DDA System

Inspired by the secure decentralized privacy system [15] and the secret sharing [14], we propose the basic decentralized privacy data aggregation on the architecture of smart contracts. In this basic scheme, we mainly use cryptographic primitives such as secret sharing to protect the security of devices' data. As shown in Figure 1, the system consists of three main entities: leaders, devices that provide data, and smart contracts.

4.1. Construction. The details of the proposed system are elaborated as follows.

(i) Initialization$(1^\lambda, \varrho, n, L.\mathrm{EOA}, D_i.\mathrm{EOA}, \mathscr{S}, \mathscr{Q})$:

(1) The leader L uses ShareGen to generate n values

$$K_i \longleftarrow \mathrm{ShareGen}\left(1^\lambda, n, \mathscr{S}\right), \quad (10)$$

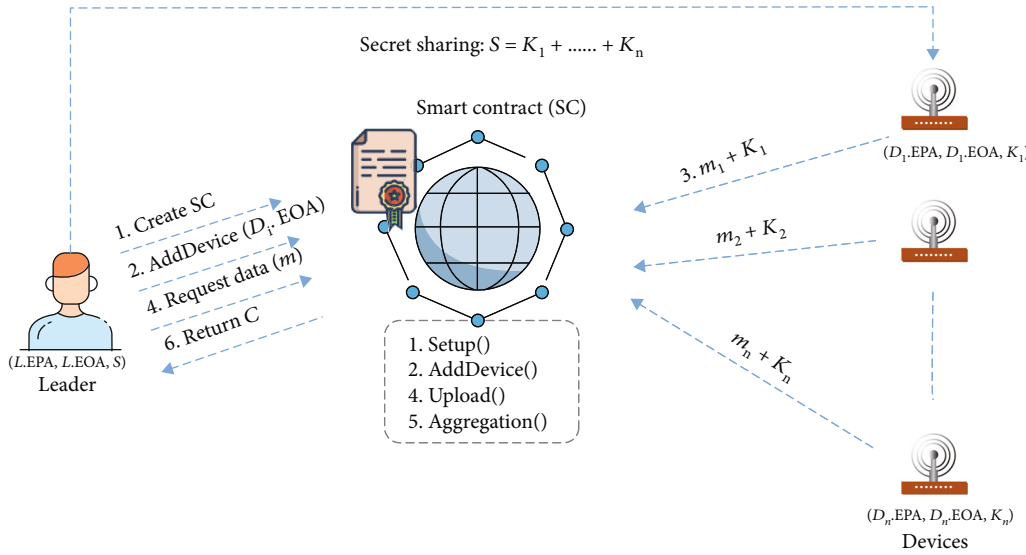

Figure 1: Sketch of basic DDA system.

and then, the leader deploys each $\mathcal{K}_i \in \{0,1\}^\ell$ to each device D_i that he/she controls. Hence, the device has parameters $(D_i.\text{EPK}, D_i.\text{EOA}, \mathcal{K}_i)$.

(2) The leader L creates SC

$$\text{SC.Address} \longleftarrow \text{InitContract}(\varrho). \tag{11}$$

And then runs AddDevice program to deploy his own devices

$$\left(\text{flag}_{L_{id}}, \mathcal{Q}'\right) \longleftarrow \text{AddDevice}(\text{SC.Address}, L.\text{EOA}, D_i.\text{EOA}, \mathcal{Q}). \tag{12}$$

(ii) DataCollection(SC.Address, L.EOA, m_i, $\mathcal{DB}$, $\mathcal{K}_i$):

(1) The leader requests data from SC. In order to protect the privacy of the raw data $m_i \in \{0,1\}^\ell$ in SC, the device D_i uses AddOnShare to encrypt data

$$DS_i \longleftarrow \text{AddOnShare}(\mathcal{K}_i, m_i). \tag{13}$$

SC receives DS_i and then updates the database $\mathcal{DB}$ by using

$$\mathcal{DB}' \longleftarrow \text{Upload}\left(\text{SC.Address}, L.\text{EOA}, \text{flag}_{L_{id}} DS_i, \mathcal{DB}\right). \tag{14}$$

(2) The leader runs Aggregation. In fact, the data will be aggregated using a leader-specified algorithm PartialRec:

$$C \longleftarrow \text{Aggregation}\left(\text{SC.Address}, \mathcal{DB}', L.\text{EOA}\right) = \text{PartialRec}\left(\mathcal{K}_1^{m_1}, \cdots, \mathcal{K}_n^{m_n}\right). \tag{15}$$

(iii) DataReconstruction$(C, \mathcal{S})$: the leader uses his secret value to recover the raw data by computing:

$$M = \text{Extract}(C, \mathcal{S}). \tag{16}$$

4.2. Security Analysis. The security proof of the proposed DDA is located on the supplemental materials. Here, we sketch the security in an implicit way with respect to SDS, NDS, SCS, and OS.

(i) (SDS) As noted earlier (Definition 5), $\mathcal{K}_i$ is a one-time key so that the proposed system is equivalent to one-time pad. $\mathcal{A}$ does not have a $\mathcal{K}_i$ key; in order to distinguish the single device's ciphertext C, it is trivial for $\mathcal{A}$ to succeed with probability 1/2 by outputting a random guess

(ii) (NDS) As noted earlier (Definition 6), the $n-1$ key K give no information at all on $\mathcal{S}$ so that the proposed system is equivalent to secret sharing. From Definition 1, the adversary can only be randomly selected from the domain $\mathbb{S}$, and the probability of success is $1/|\mathbb{S}|$

(iii) (SCS) As noted earlier (Definition 7), an external $\mathcal{A}$ needs to be authorized by the leader (AddDevice) to add their own data to the specified database (Upload). From Definition 2, unless A can destroy the security of the smart contract SCS, the probability of success is negligible

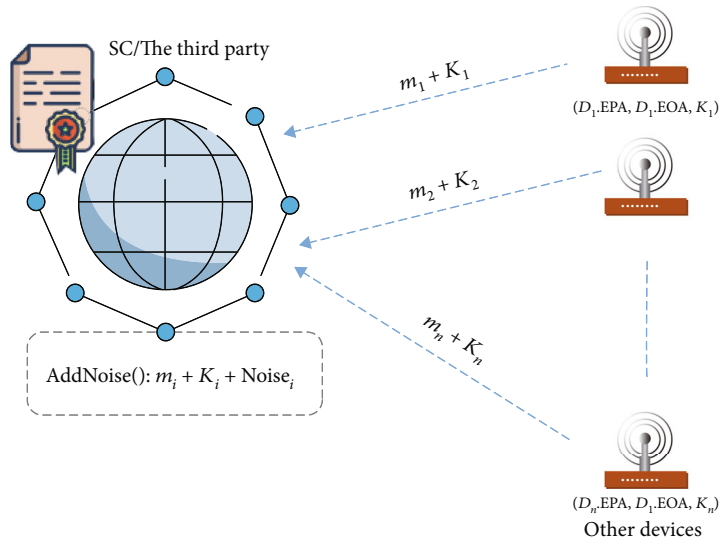

Figure 2: Sketch of centralized differential privacy DDA system.

(iv) (OS) As noted earlier (Definition 8), $\mathcal{S}$ is a one-time key so that the proposed system is equivalent to one-time pad. $\mathcal{A}$ does not have an $\mathcal{S}$ key; in order to distinguish the ciphertext C aggregation results of n devices, it is trivial for $\mathcal{A}$ to succeed with probability 1/2 by outputting a random guess

5. Full-Fledged DDA$^+$ System

In practical applications, the leaders need crowdsourcing data because more information can make better, more informed decisions for leaders. We consider if the data collected by the leader is the total number of smokers in a region. Assuming that an incoming person joins the statistics, the leader can query the number of people before and after, thereby exposing the privacy of the new user. (Assuming smoking is sensitive data for individuals.) For any such crowdsourcing, privacy preservation mechanisms should be used to reduce and control the privacy risks introduced by the data collection process [16]. We consider a balance between the usefulness of data leaking and the data collected.

Firstly, we consider a method of centralizing differential privacy (also called differential privacy on curator model). As shown in Figure 2, we can set SC or the third party to add differential privacy to the noise-adding mechanism, such as Laplace, indexing mechanism [17, 18]. However, such method cannot be realized. The specific security considerations are stated as follows:

(i) SC is currently fully open, so the noise addition process will also be exposed

(ii) Even if some of the data processing process is not public, centralized privacy may cause collusion problems

This privacy issues are actually considered in the area of differential privacy. We have some alternatives (e.g., local differential privacy) to overcome a few issues of the curator model [16].

5.1. Additional Primitives

(1) *Local Differential Privacy (LDP).* We explicitly describe the LDP settings and their concepts. In the LDP setup, there is a group of leaders, and the i-th leader has a private value v_i in a certain domain D. These leaders interact with the untrusted aggregator such that the aggregator learns statistical information about the distribution of private values in the leader population, while the information leakage for each individual is bounded. Specifically, the leader uses the algorithm $\mathcal{L}$ to perturb the private value v_i and sends $\mathcal{L}(v_i)$ to the aggregator. The aggregator then processes the collected reports to recover statistical information. The algorithm L satisfies the following properties.

(2) *Token-Controlled Public Key Encryption.* The point of a token-controlled public key encryption scheme is that the sender randomly picks a token from predefined token space. The sender encrypts the information using the token and the receiver's public key. In addition, the sender needs to send the token to a "semitrusted" third party. The receiver decrypts using the valid token and private key. We now formally define a token-controlled public key encryption scheme as follows.

Definition 9 (LDP) [19, 20]. A randomized function $\mathcal{L}$ satisfies $\in$-local differential privacy ($\in$-LDP) if and only if for any two input tuples $v, v' \in \text{Dom}(\mathcal{L})$ and for any possible output y of $\mathcal{L}$, we have

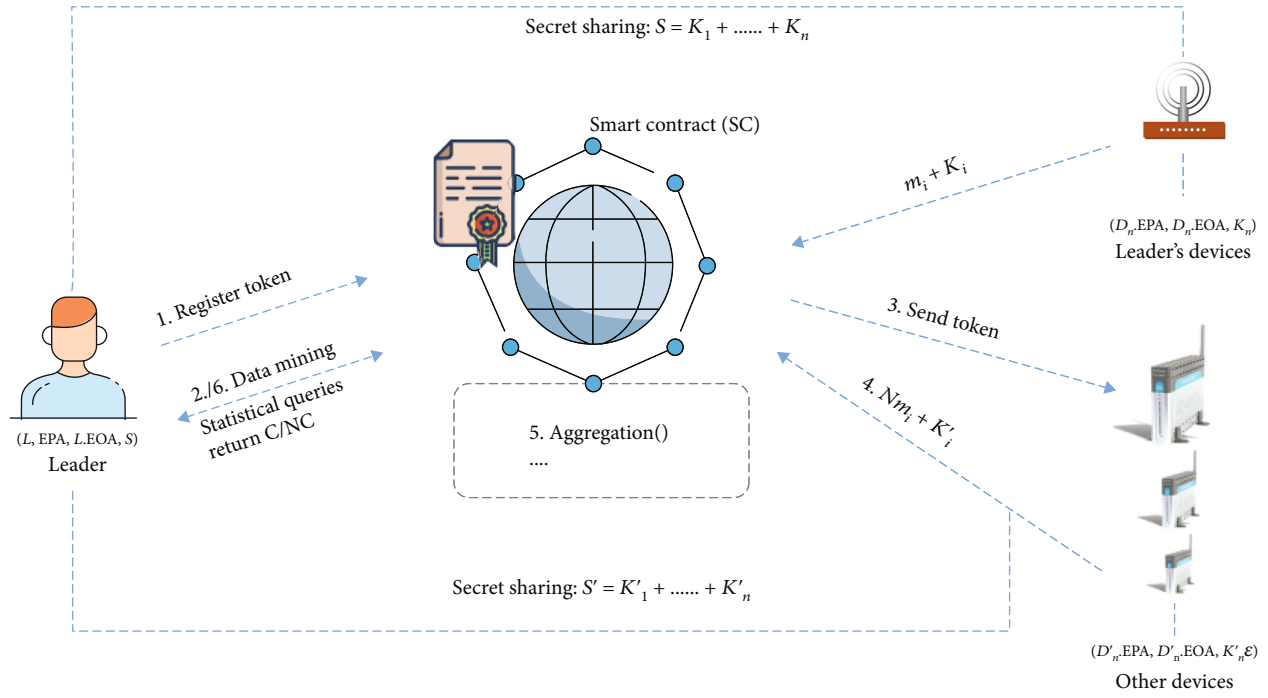

Figure 3: Sketch of local differential privacy DDA$^+$ system.

$$\Pr[\mathscr{L}(v) = y] \leq e^{\epsilon} \times \Pr\left[\mathscr{L}\left(v'\right) = y\right]. \tag{17}$$

Definition 10 (TCPKE) [21]. A token-controlled public key encryption (in short TCPKE) scheme consists of the following algorithms:

(i) $GK^{\text{TCPKE}}(1^\lambda)$: it takes a security parameter $1^\lambda \in \mathbb{N}$ as input and outputs a private and public key pair (sk, pk). Note that pk is the security parameter, finite plaintext space $\mathscr{P}$, a finite token space $\mathscr{T}$, and a ciphertext space C

(ii) $GT^{\text{TCPKE}}(1^\lambda)$: it takes a security parameter $1^\lambda \in \mathbb{N}$ as input and randomly outputs a token $k \in \mathscr{T}$

(iii) $E^{\text{TCPKE}}(\text{pk}, \text{tk}, m)$: it takes pk, tk, and $m \in \mathscr{P}$ as input and outputs a ciphertext $c \in \mathscr{C}$

(iv) $D^{\text{TCPKE}}(\text{sk}, \text{tk}, c)$: it takes sk, tk, and c as input; this algorithm outputs a plaintext $m \in \mathscr{P}$ or a special symbol $\perp$ as null

5.2. Details of DDA$^+$. Based on the above security concerns, we have modified the model.

(i) LDP [19]: the differential privacy is deployed on the local device side, and the other devices response to the leader with strong ε-differential privacy guarantees

(ii) TCPKE [21]: the cryptographic primitives (token) are added to ensure the unforgeability of the data requester and resist against external adversary attacks

As shown in Figure 3, the proposed full-fledged DDA$^+$ system consists of two parts. (1) The data interaction between the leader and the device is discussed in Section 4. (2) The leader interacts with the external devices. As a conclusion, the leader only collects randomized answers from each IoT device (by LDP techniques). We will show the details of data interaction with external devices.

5.2.1. Definition. We slightly modify the definition of the DDA model. Note that for each IoT, we assume a universal for the ease of presentation and analysis. The DDA$^+$ syntax is formally described as follows:

(i) Initialization$(1^\lambda, \varrho, n, \mathscr{S}', L.\text{EPK}, D'_i.\text{EOA}, \mathscr{NQ})$: the algorithm takes as inputs $1^\lambda, \varrho, n$, leader's private key $\mathscr{S}'$, the private address registered by the leader on a platform L.EPK, device address set $\mathscr{NQ}$, and generates K'_i, SC.Address, $\text{flag}'_{\text{Uid}}$, D'_i.pk, D'_i.sk, C, $\mathscr{NQ}'$

(ii) DataCollection (SC.Address, $\mathscr{C}$, L.EOA, D'_i.sk, m'_i, NDB, ϵ): the algorithm takes as inputs *SC.Address*, the leader's ciphertext and token $\mathscr{C}$, leader's public address L.EOA, device private key D'_i.sk, the raw data for the i-th device m'_i, device i stored database NDB, the device's privacy budget, and outputs NC

(iii) DataReconstruction (NC, S'): the algorithm takes inputs as ciphertext NC, leader's private key $\mathcal{S}'$, and outputs NM

5.2.2. Construction

(i) Initialization$(1^\lambda, \varrho, \mathrm{n}, \mathrm{S}^0, \mathrm{L.EPK}, D_i.\mathrm{EOA}, \mathcal{MQ})$:

(1) The leader L uses ShareGen to generate n values: $\mathcal{K}_i' \longleftarrow \mathrm{ShareGen}(1^\lambda, n, S')$. Note that these $\mathcal{K}_i' \in \{0,1\}^\ell$ are used by devices that are not associated with L

(2) The leader L creates SC

$$\mathrm{SC.Address} \longleftarrow \mathrm{InitContract}(\varrho) \tag{18}$$

and then runs AddDevice program to authorize alien device:

$$\left(\mathrm{flag}_{U_{\mathrm{id}}}', \mathcal{MQ}'\right) \longleftarrow \mathrm{AddDevice} \cdot \left(\mathrm{SC.Address}, L.\mathrm{EOA}, D_i'.\mathrm{EOA}, \mathcal{MQ}\right). \tag{19}$$

(3) The device D_i' runs GK^{TCPKE} algorithm and generates a private and public key pair:

$$\left(D_i'.\mathrm{pk}, D_i'.\mathrm{sk}\right) \longleftarrow GK^{\mathrm{TCPKE}}\left(1^\lambda\right). \tag{20}$$

(4) The leader runs GT^{TCPKE} and randomly chooses a token $\mathrm{tk} \in T$:

$$\mathrm{tk} \xleftarrow{\$} GT^{\mathrm{TCPKE}}\left(1^\lambda\right). \tag{21}$$

Then, the leader needs to encrypt a shared secret value K_i^0, calculated as follows:

$$c_i = E^{\mathrm{TCPKE}}\left(D_i'.\mathrm{pk}_i, \mathrm{tk}, \mathcal{K}_i'\right). \tag{22}$$

Finally, the leader releases $C = (c_i, \mathrm{tk})$. Note that tk is transmitted to SC, and SC will sign tk for the device specified by the leader, and the device will verify it.

(ii) DataCollection (SC.Address, $\mathcal{C}$, L.EOA, $D_i'.\mathrm{sk}$, m_i', NDB, ϵ):

(1) The device decrypts c_i to obtain $\mathcal{K}_i'$:

$$K_i' = D^{\mathrm{TCPKE}}\left(D_i'.\mathrm{sk}, \mathrm{tk}, c_i\right). \tag{23}$$

(2) In the local differential privacy budget ε, the device uses the algorithm L to generate the noisy data $Nm_i \in \{0,1\}^\ell$:

$$Nm_i \longleftarrow \mathscr{L}\left(m_i', \varepsilon\right). \tag{24}$$

(3) In order to protect the privacy of the noisy data in SC, the device D_i' runs the AddOnShare algorithm:

$$\mathrm{NDS}_i \longleftarrow \mathrm{AddOnShare}\left(\mathcal{K}_i', Nm_i\right). \tag{25}$$

Then, it updates the database NDB where it is located:

$$\mathrm{NDB}' \longleftarrow \mathrm{Upload}\left(\mathrm{SC.Address}, L.\mathrm{EOA}, \mathrm{flag}_{L_{\mathrm{id}}}', \mathrm{NDS}_i, \mathrm{NDB}\right). \tag{26}$$

(4) The leader runs the Aggregation program. In fact, the data will be aggregated using a leader-specified algorithm PartialRec:

$$\mathrm{NC} \longleftarrow \mathrm{Aggregation}\left(\mathrm{SC.Address}, \mathrm{NDB}', L.\mathrm{EOA}\right) = \mathrm{PartialRec}\left(\mathcal{K}_i'^{Nm_1}, \cdots, \mathcal{K}_n'^{Nm_n}\right). \tag{27}$$

(iii) DataReconstruction$(\mathrm{NC}, \mathrm{S}^0)$: the leader decrypts using his secret value to restore the noisy data by computing:

$$\mathrm{NM} = \mathrm{Extract}\left(\mathrm{NC}, \mathcal{S}'\right). \tag{28}$$

The leader can query multiple times for data statistics and mining.

5.2.3. Security Analysis. The extended DDA$^+$ system still satisfies the security of SDS, NDS, SCS, and OS. The proof of the scheme is similar to that of Section 4, and we do not discuss it further

5.2.4. Privacy Analysis. According to Definition 9, the leader who receives the perturbed tuple y cannot distinguish whether the true tuple is v or another v^0 with high confidence (controlled by parameter), regardless of the background information of the leader. This provides plausible deniability to the leader. The attributes of the tuple v_i can be either numerical or categorical. According to the privacy data types of IoT devices, different statistical methods are used for estimation and more details in Nguyen et al. [22], e.g., for a single binary attribute, it is sufficient to estimate the

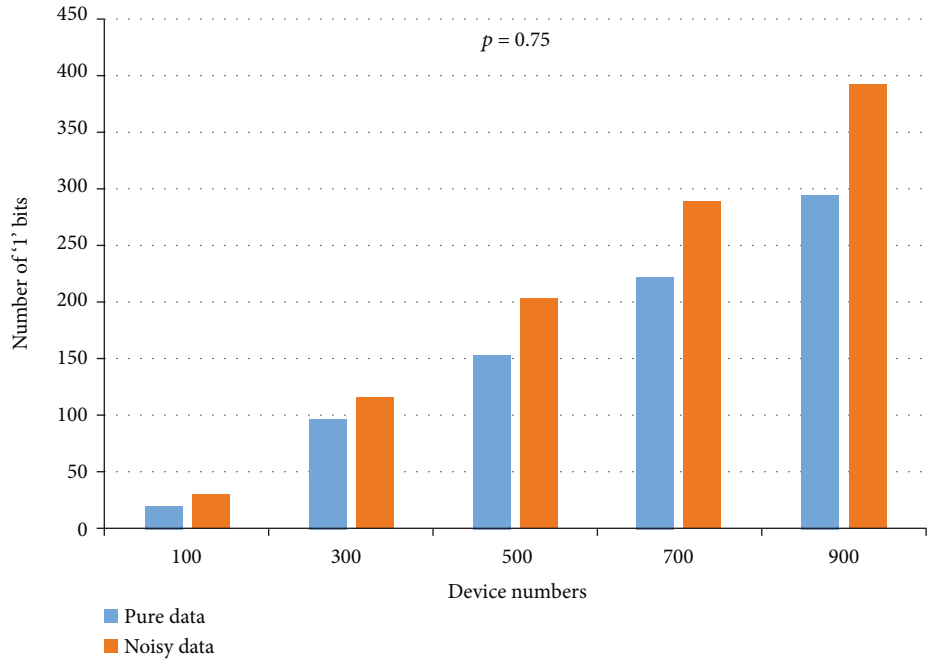

Figure 4: The deviation of pure data and noisy data with $p = 0.75$.

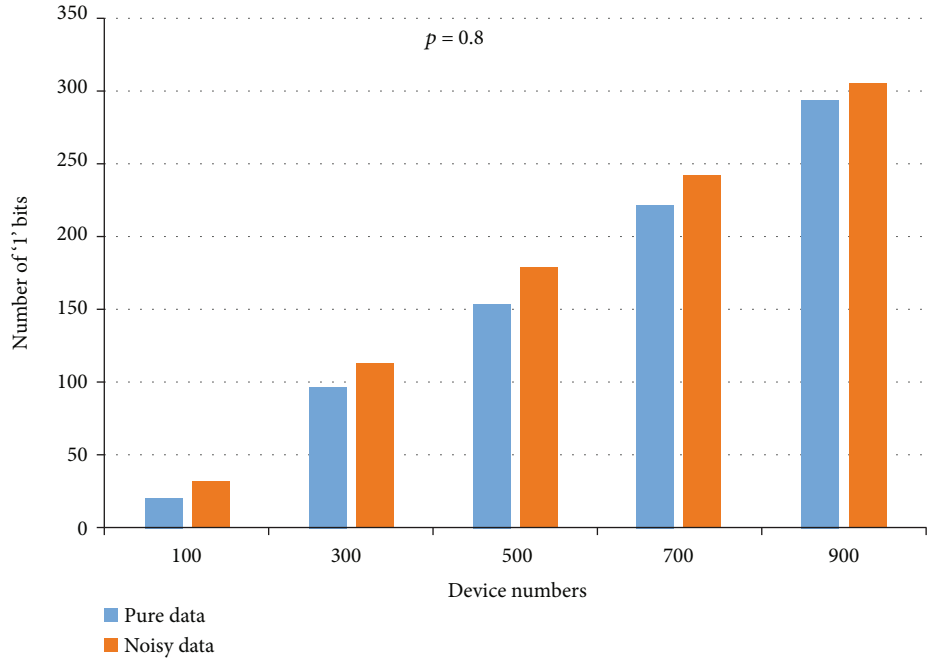

Figure 5: The deviation of pure data and noisy data with $p = 0.8$.

distribution of IoT device data using a typical random response [23]. We consider that in the following scenario, n sensors (IoT devices) are distributed among n patients, and the probability that a patient has a disease of type A is π; we want to count the expected value of π. If the aggregator (leader) directly obtains the corresponding data of the user for statistics, the patient will reveal privacy. Thus, each user u_i reports her true answer with probability p and random answer with probability $1 - p$. Then, we can simply calculate the expected value

$$E\widehat{\pi} = \frac{1}{2p-1}[p - 1 + \pi p + (1-\pi)(1-p)] = \pi. \tag{29}$$

According to Definition 9, ϵ-LDP require that $\epsilon = \ln(p/(1-p))$.

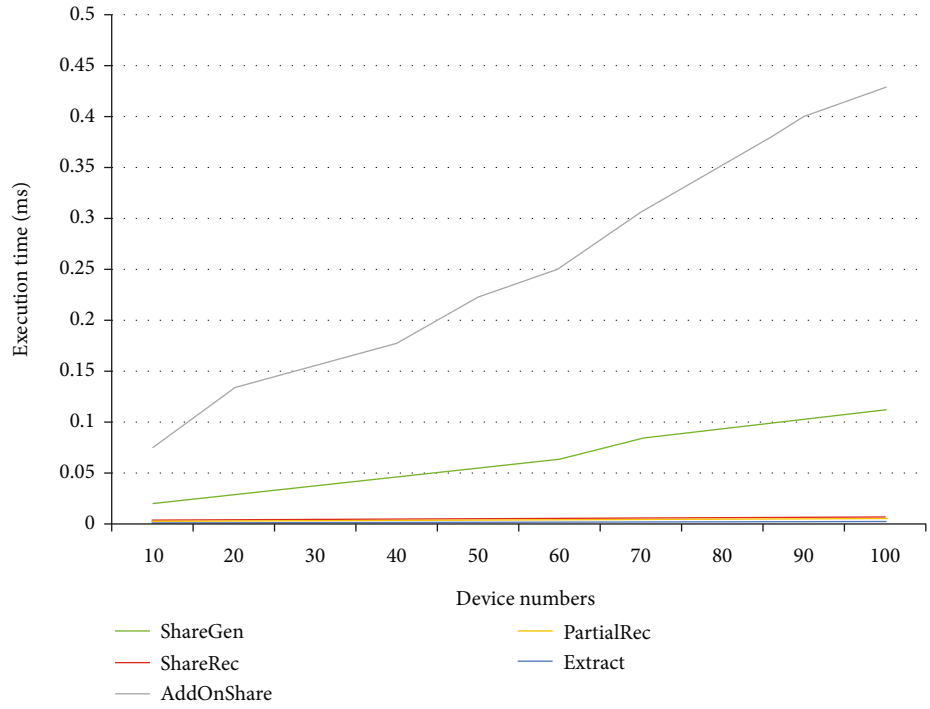

Figure 6: Performance of additive secret sharing.

Table 3: Transaction fee.

Transaction	Gas used (wei)	Transaction fee (wei)
InitContract	757658	$1.634E+15$
AddDevice	114037	$2.461E+14$
RemoveDevice	37940	$8.187E+13$
Upload	46863	$1.101E+14$
Aggregation	58249	$1.259E+14$

6. Implementations

We discuss the implementations in the aspects of LDP, secret sharing, and smart contracts.

6.1. The LDP Perturbation Data Function $\mathscr{L}$. We use Python 2.7 programming language to implement the experiments. Then, we take Raspberry Pi 3 as a data collector (the leader's device or other device) and they equip a quad-core ARMv7 CPU 1200 MHz and 1 GB RAM. Moreover, we use the Google RAPPOR [24] to realize the LDP. In DDA$^+$ system, the devices run this algorithm to generate the noisy data. Suppose each device only stores a 1-bit binary data, where the probability of the number of 1s in all the devices' data is π. Each device determines a true value (without flipping the 1-bit data) with a probability p. After the randomized response, we can obtain the noisy data. For analyzing the deviation from the original and noisy data aggregations, we fix $\pi = 0.3$ and then set $p = 0.75$ and $p = 0.8$ to get $\epsilon = \ln(3)$ and $\epsilon = \ln(4)$, respectively. The deviation of pure data and noisy data with different p is shown in Figures 4 and 5. By observation over the experimental results, it suffices to use $p = 0.75$ if we accept 10% deviation on 300 devices ($p = 0.8$ if on more than 500 devices).

6.2. Performance of the Secret Sharing Scheme. For evaluating the performance, we use python programming language with version 2.7 and import random module, numpy module, etc., to implement the experiment, running Intel(R) Core(TM) i7-7700 CPU 3.60 GHz and 8 GB memory in Windows 10 as the operating environment of Ethereum. Firstly, the leader uses ShareGen to generate shares and sends them to devices. Devices use AddOnShare to encrypt the data. Next, as long as the leader wants to require some data, he uses PartialRec and Extract to recover the data. The execution time of each algorithm is shown in Figure 6.

6.3. Simulations on Smart Contracts. We run our DDA$^+$ system on the Ethereum blockchain through Solidity 0.6.12 and deploy it upon the Ropsten test network on February 17, 2021, where the gas price is approximately 2.1578 Gwei. Note that, at the same time, Ethereum average gas price on the public network is 183.087 Gwei. On our simulation, we firstly have to deploy the main contract, where we use AddDevice and RemoveDevice transaction to add device and delete device, respectively. If the leader would like to access, he can send Aggregation transaction. When devices collect some data, they send Upload transaction to the blockchain. In Table 3, every transaction has a different processing fee. We found that the highest cost is exactly on contract deploying. The other types of transactions only require a lesser cost.

7. Conclusions

In this paper, we propose the DDA/DDA$^+$ systems based on smart contract as the heart to achieve decentralization. However, they rely on privacy and cryptographic algorithms to protect the data of IoT devices and satisfy the security requirements. To wrap up our techniques, we use secret sharing to preserve the data processing mechanism efficiently. We apply local differential privacy and tokens as extensions of the access capabilities of other leaders' devices.

Data Availability

Data sharing not applicable—no new data generated, or the article describes entirely theoretical research.

Disclosure

The earlier version of this work was published in DSC 2021.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported in part by the Ministry of Science and Technology of Taiwan (Nos. 106-2218-E-155-008-MY3 and 109-2628-E-155-001-MY3). We thank Zhong-Yi Guo and Yunmin He (YZU) for initial discussions.

References

[1] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "IoT security: ongoing challenges and research opportunities," in *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pp. 230–234, Matsue, Japan, 2014.

[2] Y.-A. De Montjoye, E. Shmueli, S. S. Wang, and A. S. Pentland, "openpds: protecting the privacy of metadata through safeanswers," *PLoS One*, vol. 9, no. 7, article e98790, 2014.

[3] D. C. Plummer, M. Reynolds, C. S. Golvin et al., "Top strategic predictions for 2017 and beyond: surviving the storm winds of digital disruption," *Gartner report G00315910*, 2017.

[4] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *International conference on the theory and applications of cryptographic techniques*, pp. 506–522, Springer, 2004.

[5] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 169–178, Bethesda, MD, USA, 2009.

[6] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[7] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, Manubot, 2008.

[8] M. Abramaowicz, "Cryptocurrency-based law," *Ariz. L. Rev.*, vol. 58, p. 359, 2016.

[9] L. Zhou, L. Wang, Y. Sun, and P. Lv, "Beekeeper: a blockchain-based IoT system with secure storage and homomorphic computation," *IEEE Access*, vol. 6, pp. 43472–43488, 2018.

[10] L. Zhou, L. Wang, T. Ai, and Y. Sun, "Beekeeper 2.0: confidential blockchain-enabled IoT system with fully homomorphic computation," *Sensors*, vol. 18, no. 11, p. 3785, 2018.

[11] C.-M. Chen, X. Deng, W. Gan, J. Chen, and S. K. H. Islam, "A secure blockchain-based group key agreement protocol for IoT," *The Journal of Supercomputing*, pp. 1–23, 2021.

[12] X. Xie and Y.-C. Chen, "Decentralized data aggregation: a new secure framework based on lightweight cryptographic algorithms," in *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, pp. 1-2, Aizuwakamatsu, Fukushima, Japan, 2021.

[13] J. Benaloh and J. Leichter, "Generalized secret sharing and monotone functions," in *Proceedings on Advances in cryptology*, pp. 27–35, Springer, 1990.

[14] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[15] G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy: using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops*, pp. 180–184, San Jose, CA, USA, 2015.

[16] M. U. Hassan, M. H. Rehmani, and J. Chen, "Differential privacy in blockchain technology: a futuristic approach," *Journal of Parallel and Distributed Computing*, vol. 145, pp. 50–74, 2020.

[17] C. Dwork, "Differential privacy," *Encyclopedia of Cryptography and Security*, pp. 338–340, 2011.

[18] C. Lin, P. Wang, H. Song, Y. Zhou, Q. Liu, and G. Wu, "A differential privacy protection scheme for sensitive big data in body sensor networks," *Annals of Telecommunications*, vol. 71, no. 9-10, pp. 465–475, 2016.

[19] G. Cormode, S. Jha, T. Kulkarni, N. Li, D. Srivastava, and T. Wang, "Privacy at scale: local differential privacy in practice," in *Proceedings of the 2018 International Conference on Management of Data*, pp. 1655–1658, Houston, TX, USA, 2018.

[20] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 729–745, Vancouver, BC, Canada, 2017.

[21] J. Baek, R. Safavi-Naini, and W. Susilo, "Tokencontrolled public key encryption," in *International Conference on Information Security Practice and Experience*, pp. 386–397, Springer, 2005.

[22] T. T. Nguyên, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin, "Collecting and analyzing data from smart device users with local differential privacy," 2016, http://arxiv.org/abs/1606.05053.

[23] S. L. Warner, "Randomized response: a survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.

[24] U. Erlingsson, V. Pihur, and A. Korolova, "Rappor: randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 1054–1067, Scottsdale, AZ, USA, 2014.