Achieving Accountable and Efficient Data Sharing in Industrial Internet of Things

Cheng Huang, *Student Member, IEEE*, Dongxiao Liu, *Student Member, IEEE*, Jianbing Ni, *Member, IEEE*, Rongxing Lu, *Senior Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—In this article, we propose an accountable and efficient data sharing scheme for industrial IoT (IIoT), named an accountable and data sharing scheme (ADS), in which a data owner can pursue the responsibility of a data receiver if the latter leaks some sensitive shared data to the public for profits while without permission (i.e., accountability). Specifically, ADS is built upon an adaptive decentralized oblivious transfer protocol together with a zero-knowledge proof technique, which enables the data receiver's private key to be hidden from the data owner and yet correctly embedded into the shared data during the process of data sharing. Once data breaches occur, the private key can be automatically revealed to the data owner so as to achieve the accountability. In addition, with ADS, a group of sharing providers can also assist IIoT devices in handling heavy computational tasks via the secret sharing technique without sacrificing the security. Extensive performance evaluations are conducted, and the simulation results demonstrate that ADS has high computational efficiency, making it well fit for IIoT.**

***Index Terms*—Accountability, data sharing, data breach, industrial Internet of Things (IIoT).**

I. Introduction

INDUSTRIAL Internet of Things (IIoT), which indicates a large potential in advancing the development of many industrial systems such as energy, manufacturing, logistics, and transportation [1], [2], has recently attracted considerable attention. By leveraging the highly-connected smart IIoT devices to sense, monitor, collect, exchange, and analyze data generated in the industrial process, IIoT can assist in building comprehensive and intelligent solutions in the industrial systems [3], [4]. For example, IIoT enables logistics companies to easily track their drivers' activities, vehicles' locations, and all goods' delivery status through a real-time logistics management system supported by heterogeneous networks and wide-deployed IIoT devices. Once goods are delivered to or arrived at a certain place, e.g., a warehouse, an instant notification can be given in the system, which achieves both transparency and traceability in logistics.

Essentially, IIoT inherently incorporates two main characteristics: interconnection and interoperability [5], [6]. Interconnection means IIoT devices are connected with each other via ubiquitous networks with high-speed transmission; while interoperability means these devices exchange information and make use of the information to perform data analytics [7]. In other words, seamless data sharing among various industrial entities and their systems, e.g., manufacturers, suppliers, and carriers, is required to break the data isolation in IIoT, which also promotes the system-level decision making for a wide range of industry sectors [8]. For instance, when logistics companies obtain the shared real-time traffic data and road conditions from transportation companies, they can optimize the schedule of delivering goods to shorten the delivery time and save the delivery cost prominently.

Despite its promising features, there still exists an inevitable security issue that hinders open data sharing in IIoT, i.e., a data owner (sender) cannot control the access of shared data after they have been shared with many data receivers. In most of the data sharing cases, the sender only allows the receiver to utilize the shared data but does not allow a receiver to leak the shared data to others or the public for profits or other self-interest purposes without permission. If an event of data leakage happens (e.g., the sender is aware of the data breach and acquires the leaked data from the Internet), no matter intentionally or unintentionally, the corresponding receiver who leaks the data should be tracked and be responsible for the event. This issue is referred to as the accountability issue.

To achieve the accountability in data sharing, three requirements should be satisfied. First, some evidence should be included in the shared data and should be explicitly extracted after the data breach occurs. Second, there should be an authority, who keeps online to collect the evidence and to pursue the responsibility of the receiver who leaks the shared data if necessary. Third, cumbersome computational costs on IIoT devices should not be permitted as they are always resource-limited. Nevertheless, these requirements cannot be easily achieved simultaneously. For the first two requirements, the classical watermarking techniques [9], [10] could be applied, since they

Manuscript received February 25, 2020; accepted March 16, 2020. Date of publication March 30, 2020; date of current version November 18, 2020. Paper no. TII-20-0996. *(Corresponding author: Jianbing Ni.)*

Cheng Huang, Dongxiao Liu, and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: c225huan@uwaterloo.ca; dongxiao.liu@uwaterloo.ca; sshen@uwaterloo.ca).

Jianbing Ni is with the Department of Electrical and Computing Engineering, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: jianbing.ni@queensu.ca).

Rongxing Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: rlu1@unb.ca).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2020.2982942

can be used to watermark the shared data as the evidence, and a public blockchain [11] can be deployed as a decentralized authority to achieve the accountability, as shown in the recent work by Mangipudi *et al.* [12]. Although their scheme is highly creative, directly applying it into IIoT may suffer from the efficiency bottleneck. This is mainly due to the fact that it relies on the time-consuming exponential operations and public-key operations.

To address the abovementioned challenge, in this article, we propose a novel accountable and efficient data sharing scheme for IIoT, named an accountable and data sharing scheme (ADS), based on the watermarking techniques [10], [13] and several cryptographic primitives, such as distributed oblivious transfer [14] and zero-knowledge proof [15]. ADS is constructed on a multiprovider model, where the data owner (sender) shares the watermarked data blocks with the data receiver through a group of sharing providers [3]. In ADS, the receiver's private key is concealed from the sender but correctly embedded into the watermarked data blocks in the process of data sharing. When these data blocks are leaked, the sender can extract the private key of the data leaker from the data blocks, which can be used for a posterior investigation or a dispute resolution, e.g., the data leaker can be penalized via a claim-or-refund smart contract deployed on the public blockchain [11]. In addition, most of the heavy computational tasks are accomplished by the decentralized sharing providers in ADS and the computational overhead of IIoT devices can be significantly reduced. In summary, the contribution of this article is four fold.

1) ADS is proposed under the semihonest adversary model. Based on an adaptive distributed oblivious transfer protocol together with a zero-knowledge proof technique, ADS allows a data owner and a data receiver to achieve accountable data sharing through a group of sharing providers. These sharing providers assist IIoT devices in handling heavy computational tasks, which makes ADS highly efficient in terms of computational costs.
2) To improve the effectiveness of accountability, R-ADS is introduced based on ADS, by making use of public-key homomorphic encryption, permutation, and randomization techniques. R-ADS ensures that, even if the data receiver only leaks partial shared data but not the whole shared data, the data owner can still trace his/her responsibility by tracing his/her key.
3) To deal with malicious sharing providers, R-ADS is further extended to E-ADS by utilizing the zero-knowledge proof to verify the correctness of the sharing providers' operations. E-ADS ensures that, even if a small portion of sharing providers are malicious to perform arbitrary operations in the process of data sharing, such malicious behavior can still be detected through the public verification and the accountability is still achieved.
4) Security analysis demonstrates that desirable security goals are achieved in ADS, R-ADS, and E-ADS. In addition, the simulation results also demonstrate that ADS is efficient in comparison with the existing schemes [12], [16], and R-ADS and E-ADS can provide a stronger security guarantee at the expense of acceptable computational costs.

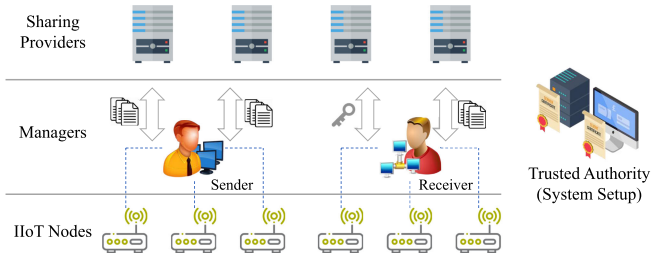


Fig. 1. System model.

The remainder of this article is organized as follows. In Section II, we introduce the system model, security requirements, and design goals. Then, we propose ADS in Section III. Subsequently, security analysis and performance evaluation are shown in Sections IV and V, respectively. Finally, Section VI reviews some related works and Section VII concludes this article.

II. Model and Design Goal

In this section, we first sketch the system model of data sharing in IIoT, and then we present the security requirements and the design goals.

A. System Model

In the system model, there exist four types of entities, namely IIoT Nodes, Managers, Sharing Providers, and Trusted Authority, as shown in Fig. 1.

1) IIoT nodes: IIoT nodes involve different kinds of sensors, devices, and machines with low computational power. They are usually deployed in the industrial process, generating a variety of data, and storing them temporarily. If necessary, they can upload some data to the manager for a long-term storage.
2) Managers: There exist some managers who control and manage their IIoT nodes via commands. They also store the long-term data received from their IIoT nodes.
3) Sharing providers (SPs): A group of sharing providers, which are highly available and well-provisioned servers with the strong computational capability and sufficient resources for communication and storage.
4) Trusted authority (TA): A trusted authority is fully trusted and is responsible for generating the public parameters and initializing the system.

In the system model, when a manager $\mathcal{M}_1$ (receiver) needs to obtain the data of an IIoT node belonging to another manager $\mathcal{M}_2$ (sender), she first sends a data sharing request to $\mathcal{M}_2$. After $\mathcal{M}_2$ authorizes the request, $\mathcal{M}_2$ (sender) collects the data from the corresponding IIoT node and shares the data with $\mathcal{M}_1$ (receiver) through a group of SPs. In the process of data sharing, the data of $\mathcal{M}_2$ is shared with $\mathcal{M}_1$, while the private key of $\mathcal{M}_1$ (corresponding to his/her public key) is embedded into the share

data in a hidden way, such that $\mathcal{M}_1$ can be held accountable for his/her behavior.

B. Security Requirements

ADS requires the availability of private and authenticated communication channels between IIoT nodes and managers, managers and SPs, among SPs. The network is required to be synchronous and these communication channels are secure, i.e., the communications cannot be eavesdropped and be tampered with. Most of the SPs are assumed to be honest, and at most $k-1$ out of n ($2k-1 \leq n$ and n is the number of SPs) SPs can be corrupted and collude with each other to behave in a semihonest manner (Adversary $\mathcal{A}_1$) or in an arbitrary/malicious manner (Adversary $\mathcal{A}_2$). The set of corrupted SPs is not known in advance, but it is a static corruption and does not change during scheme execution. In practice, each SP is operated independently, i.e., each SP is maintained by a separate operator to limit the risks of the SPs being compromised or colluding with each other. Moreover, both the sender and the receiver are assumed to be semihonest and wish to complete the scheme to allow the receiver to obtain the shared data, and there exist some robust watermarking techniques for numeric and non-numeric data generated by IIoT nodes [10], [13], i.e., two properties should be guaranteed: 1) the watermark can be easily detected by the sender; and 2) the watermark cannot be changed or removed from the data block once added. In this situation, the following security requirements should be satisfied.

1) Data confidentiality ($\mathcal{A}_1$ and $\mathcal{A}_2$):
   a) Up to $k-1$ out of n collusive SPs cannot obtain the shared data of the sender nor the private key of the receiver.
   b) The sender, even though colluding with up to $k-1$ SPs, cannot obtain the private key of receiver before the disclosure of the shared data by the receiver.
   c) The receiver, even though colluding with up to $k-1$ SPs, cannot obtain the shared data before the end of data sharing and her private key is correctly embedded.
2) Data accountability ($\mathcal{A}_1$ and $\mathcal{A}_2$): In case of unauthorized disclosure of the shared data by the receiver, the private key of the receiver is revealed to the sender, which can be utilized to trace the responsibility.
3) Data correctness ($\mathcal{A}_2$): After up to $k-1$ SPs corrupt and behave maliciously without following the scheme, the sender can still share the shared data and the receiver can correctly receive the data. In the meantime, the malicious behavior of these malicious SPs can be detected.

C. Design Goals

Under the aforementioned system model and security requirements, our goal is to propose an accountable and efficient data sharing scheme for IIoT, and the following two design objectives should be achieved.

1) Security: The security requirements mentioned previously should be satisfied in ADS. Namely, data confidentiality, data accountability, and data correctness should be achieved.
2) Efficiency: ADS should be lightweight. To implement the data sharing scheme for a real-world IIoT system, the computational efficiency should be considered.

III. Proposed Scheme

In this section, we first review several cryptographic primitives, and then propose an ADS and its variants, including R-ADS and E-ADS, based on these primitives.

A. Preliminaries

1) Oblivious Transfer: A 1 out of 2 oblivious transfer (OT) is a two-party (a sender and a receiver) protocol. In the protocol, the sender has two messages Msg_0 and Msg_1 and the receiver has a bit $b \in \{0,1\}$. The sender aims to transfer Msg_b to the receiver. At the end of the protocol, and the receiver does not learn any information about Msg_{1-b} and the sender does not learn b. We consider a distributed version, where the oblivious transfer is achieved by multiple distributed parties. For more details, please refer to [14].

2) Secret Sharing: A k out of n secret sharing protocol allows a dealer to share its secret s with a set of players $P = \{P_1, P_2, \ldots, P_N\}$. Each player P_i can obtain and store a secret s_i. If more than k out of n players reveal their secrets s_i, the secret s can be reconstructed.

3) Zero-Knowledge Proof: The zero-knowledge proof of knowledge allows the prover to generate a cryptographic proof with a corresponding statement, and the verifier can verify the proof to check the correctness of the statement. Generally, a zero knowledge proof of knowledge can be expressed in a particular notation. For instance, ZkPoK$\{(a) : A = g_1^a\}$ denotes "zero-knowledge proof of knowledge of a such that $A = g_1^a$ holds". The Fiat–Shamir heuristic can be applied to turn the abovementioned interactive zero-knowledge proof of knowledge into the noninteractive one in the random oracle model. For more details, see [15].

4) Decision Diffie–Hellman Problem: Let G_q be a cyclic multiplicative group with the prime order q. Let g is a generator of G_q. Let a, b, and z be drawn from the uniform distribution on Z_q^*. The advantage of a probabilistic algorithm $\mathcal{A}$ in solving the decisional Diffie–Hellman problem in G_q (DDH problem), $\mathrm{Adv}_{\mathcal{A},G}^{\mathrm{DDH}}(\kappa) = |Pr[\mathcal{A}(g, g^a, g^b, g^{ab})] - Pr[\mathcal{A}(g, g^a, g^b, g^z)]$, is negligible in the security parameter κ.

B. Details of ADS

High-level overview: ADS is mainly constructed on the watermarking techniques [10], [13] and the distributed 1-out-of-2 OT protocol [14], as shown in Fig. 2. At the beginning of data sharing, the sender prepares two distinct watermarked data blocks, including the same shared data but with different watermarks, based on the watermarking technique. The receiver then obtains one of these two blocks based on the distributed 1-out-of-2 OT protocol but hides her choices. Apparently, to achieve the accountability, the receiver needs to convince the

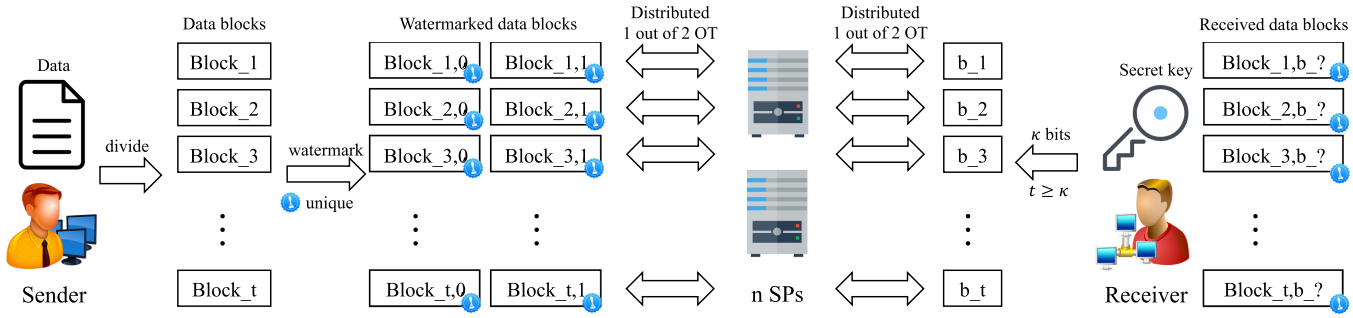


Fig. 2. High-level overview of ADS.

sender the statement that the choices are the same as the bits of the receiver's private key. Afterwards, if the shared data are leaked, the sender can identify the watermarks to extract the private key of the receiver. Therefore, directly applying the OT protocol is not useful, and the zero-knowledge proof technique [15] is integrated with the original distributed OT protocol in ADS.

Specifically, ADS includes four phases: *System setup*, *authorization*, *data sharing*, and *key tracing*.

1) *System Setup:* TA chooses a large prime p ($p = 2q + 1$ and q is a prime as well) whose length is κ, i.e., $|p| = \kappa$. TA, then, chooses two distinct generators $g \in Z_p^*$ and $h \in Z_p^*$ with order q for the multiplicative cyclic group $G_q \subset Z_p^*$. A symmetric-key encryption/decryption algorithm (the symmetric key is η bits) is chosen by TA as $\mathsf{Enc}()/\mathsf{Dec}()$, a public-key signature/verification algorithm (the public and private key are in the field of G_q and Z_q^*, respectively) is chosen by TA as $\mathsf{Sig}()/\mathsf{Ver}()$, and a cryptographic hash function is chosen by TA as $\mathsf{H}_1() : \{0,1\}^* \rightarrow \{0,1\}^\eta$. Finally, TA publishes the public parameters params

$$\mathsf{params} = \{p, q, g, h, Z_p^*, Z_q^*, G_q, \mathsf{Enc}()/\mathsf{Dec}(), \mathsf{Sig}()/\mathsf{Ver}(), \mathsf{H}_1()\}.$$

After parameter generation, TA initializes the system by generating and distributing the key pairs for managers (i.e., senders and receivers). TA chooses the private signature key $\mathsf{ssk} \in_R Z_q^*$ for the sender and publishes her public signature key $\mathsf{spk} = h^{\mathsf{ssk}}$. Similarly, TA chooses the private signature key $\mathsf{rsk} \in_R Z_q^*$ for the receiver and publishes her public signature key $\mathsf{rpk} = h^{\mathsf{rsk}}$. TA also sets the threshold (k, n), which represents that n SPs are deployed and at most $k-1$ SPs are allowed to be compromised. According to the size of the shared data, the shared data can be uniformly divided into t data blocks. When TA sets $t = \kappa$, the setting means the private signature key rsk is embedded into the shared data once. In general, TA sets $t = \epsilon \times \kappa$ where ϵ is a positive integer number except 0. After the initialization, TA is offline in the system.

2) *Authorization:* A receiver first sends the data sharing request Req to the sender. If the sender agrees to share the data, the sender signs the request by running $\sigma = \mathsf{Sig}_{\mathsf{ssk}}(\mathrm{Req})$ and broadcasts the signature σ and the request Req to n SPs. Each SP verifies the sender's signature σ by running $\mathsf{Ver}_{\mathsf{spk}}(\sigma, \mathrm{Req})$. If it passes the verification, each SP creates a separate sharing pool for the sender and the receiver. In other words, data sharing can occur in parallel.

3) *Data Sharing:* This phase involves four subphases: *Data preparation*, *data encryption*, *data transfer*, and *data decryption*.

a) *Data preparation:* The sender divides the shared data M into t data blocks $\{M_1, M_2, \ldots, M_t\}$ and for each block M_i ($i = 1$ to t), she generates two unique watermarked blocks $\{M_{0,i}, M_{1,i}\}$. Then, she randomly chooses $2t$ group elements $\{z_{0,i} \in_R G_q, z_{1,i} \in_R G_q\}_{i=1}^t$ and encrypts the blocks as $E_{0,i} = \mathsf{Enc}_{\mathsf{H}_1(z_{0,i})}(M_{0,i})$ and $E_{1,i} = \mathsf{Enc}_{\mathsf{H}_1(z_{1,i})}(M_{1,i})$ for $i = 1$ to t. The symmetric keys used for encryption are $\mathsf{H}_1(z_{0,i})$ and $\mathsf{H}_1(z_{1,i})$. Finally, she sends $\{E_{0,i}, E_{1,i}\}_{i=1}^t$ to the receiver.

b) *Data encryption:* The sender runs $\{j, f(j)_{0,i}\}_{j=1}^n = SS(\mathsf{params}, k, n, z_{0,i})$ and $\{j, f(j)_{1,i}\}_{j=1}^n = SS(\mathsf{params}, k, n, z_{1,i})$ for $i = 1$ to t where $SS()$ is the secret sharing algorithm, as shown in Algorithm 1. Afterwards, she distributes the shared secret $s_{i,j} = \{f(j)_{0,i}, f(j)_{1,i}\}$ to the jth SP for $j = 1$ to n. Finally, the sender notifies the receiver that the shared data can be downloaded from SPs.

c) *Data transfer:* The receiver uses the bits $\{b_1, b_2, \ldots, b_t\}$ to represent the private key rsk and chooses the random number $r_i \in_R Z_q^*$ for $i = 1$ to t (r_i should be stored temporarily and can be deleted at the end of the subphase). Then, she generates the commitment $\{y_i = g^{r_i} h^{b_i}\}_{i=1}^t$ and calculates the auxiliary information $A = g^{\sum_{i=1}^t 2^i r_i}$. Next, she performs a noninteractive zero-knowledge proof as follows to generate the proof π_{key}, randomly chooses k SPs, and sends π_{key} to these SPs

$$\mathrm{ZkPoK}\{(\{r_i, b_i\}_{i=1}^t) : \{y_i = g^{r_i} h^{b_i}\}_{i=1}^t \wedge A = g^{\sum_{i=1}^t 2^i r_i}\}.$$

Each SP verifies π_{key}. If the proof is correct, the jth SP ($j = 1$ to k) checks whether the bits of the receiver's private key are included in $\{y_i\}_{i=1}^t$ by the following

Algorithm 1: Secret Sharing $SS()$.

Input: params, k, n, s.
Output: $\{j, f(j)\}_{j=1}^{n}$.
1: Randomly pick $a_1 \in_R Z_p^*$, $a_2 \in_R Z_p^*$, ..., $a_{k-1} \in_R Z_p^*$.
2: Construct
$f(x) = s + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1} \bmod p$.
3: **for** $j = 1$ to n **do**
4: Output $\{j, f(j)\}$.
5: **end for**

equation:

$$\text{rpk} \cdot A = \prod_{i=1}^{t} (y_i)^{2^i}.$$

Then, the jth SP creates the corresponding shares as $\{c_{0,i,j}, c_{1,i,j}\}_{i=1}^{t}$ and sends them to the receiver, where $\delta_{0,i,j} \in_R Z_q^*$ and $\delta_{1,i,j} \in_R Z_q^*$

$$c_{0,i,j} = \{g^{\delta_{0,i,j}}, f(j)_{0,i} \cdot (y_i / h^0)^{\delta_{0,i,j}}\}$$
$$c_{1,i,j} = \{g^{\delta_{1,i,j}}, f(j)_{1,i} \cdot (y_i / h^1)^{\delta_{1,i,j}}\}.$$

The receiver runs the reconstruction algorithm, as shown in Algorithm 2, to restore $\{z_{b_i,i}\}_{i=1}^{t}$ as follows

$$\{z_{b_i,i}\}_{i=1}^{t} = RC(\quad \text{params}, \{c_{0,i,j}, c_{1,i,j}\}_{i=1}^{t}, \{r_i\}_{i=1}^{t}, b_i).$$

d) *Data decryption:* The receiver decrypts the encrypted data blocks as $M_{b_i,i} = \text{Dec}_{\text{H}_1(z_{b_i,i})}(E_{b_i,i})$ for $i = 1$ to t.

4) *Key Tracing:* Once the data breach occurs, that is, $\{M_{b_i,i}\}_{i=1}^{t}$ are published, the sender matches the stored watermarks to identify the choices (the bits included in the private key) of the receiver and pursues the responsibility of the receiver by submitting the private key to a posterior investigation or a dispute resolution. We omit the details of a dispute resolution and only give a short explanation. Before data sharing, the sender and receiver can create a claim-or-refund smart contract on the public blockchain where a money amount is deposited that can be spent at any time with a jointly signature of the sender and the receiver. If the private key of the receiver that signs the smart contract is revealed to the sender due to the data breach by the receiver for some self-interests purposes, the receiver is penalized automatically as the money is gained by the sender.
Note that, key tracing in ADS suffers from a special attack where the receiver only exposes partial shared data blocks $M' \subset \{M_{b_i,i}\}_{i=1}^{t}$, and thus, the sender cannot obtain all valid bits of the receiver's private key. Assuming that the receiver leaks $t' = \kappa' * \epsilon$ data blocks, the least number of valid bits leaked by the receiver is κ', and the probability that the sender can successfully guess the private key rsk is $\text{Pr}(\text{Succ}) = (\frac{1}{2})^{\kappa - \kappa'}$.

Algorithm 2: Reconstruction $RC()$.

Input: params, $\{c_{0,i,j}, c_{1,i,j}\}_{i=1}^{t}$, $\{r_i\}_{i=1}^{t}$, b_i.
Output: $\{w_{b_i,i}\}_{i=1}^{t}$.
1: Create an empty list $list = \{\}$.
2: **for** $i = 1$ to t **do**
3: **for** $j = 1$ to k **do**
4: Calculate $v_{b_i,i,j} = \frac{f(j)_{b_i,i} \cdot (y_i / h^{b_i})^{\delta_{b_i,i,j}}}{(g^{\delta_{b_i,i,j}})^{r_i}}$.
5: **end for**
6: Calculate $w_{b_i,i} = \sum_{j=1}^{k} v_{b_i,i,j} \cdot (\prod_{l=0, l \neq i}^{k} \frac{l}{l-i})$.
7: Add $w_{b_i,i}$ into $list$.
8: **end for**
9: Output $list = \{w_{b_i,i}\}_{i=1}^{t}$.

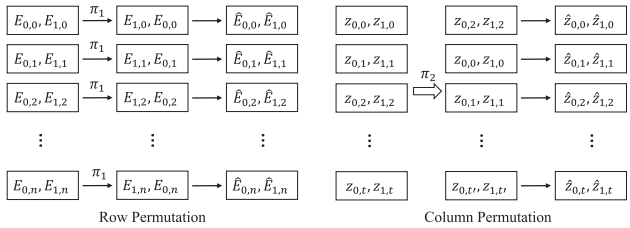


Fig. 3. Permutation operations in R-ADS.

C. Details of R-ADS

In this section, to improve the success probability of key tracing, we propose a variant of ADS, named R-ADS. Specifically, different from ADS, the receiver cannot know the correspondence between his/her choices and the watermarked data blocks in R-ADS. The basic idea behind R-ADS is called oblivious permutation. Namely, before the sender and the receiver run the distributed OT protocol to share the data blocks, the sender permutes the data blocks; after the receiver obtain the permuted data blocks, he can obliviously perform reverse permutation to recover the data blocks with the assistance of the sender. Since the permutation is oblivious to the receiver, he cannot identify, which part of the private key is revealed. The followings are the details of R-ADS. We only emphasize the differences between ADS and R-ADS, and ignore the repetitive parts.

1) *System Setup:* A public-key additively homomorphic encryption/decryption algorithm (the public and private key are in the field of G_q and Z_q^*, respectively) is chosen by TA as $Enc'()/Dec'()$ and is published as the public parameter. In addition, TA chooses a new public/private key pair $(\text{rek} = g^{\text{rdk}}, \text{rdk} \in_R Z_q^*)$ for the receiver.
2) *Data Sharing:* The subphases, including *data preparation*, *data encryption*, and *data decryption*, are different from the corresponding subphases in ADS.
a) *Data preparation:* The sender permutes $\{E_{0,i}, E_{1,i}\}$ to $\{\hat{E}_{0,i}, \hat{E}_{1,i}\}$ (row permutation π_1) for $i = 1$ to t and then permutes $\{z_{0,i}, z_{1,i}\}$ to $\{\hat{z}_{0,i}, \hat{z}_{1,i}\}$ for $i = 1$ to t (column permutation π_2), as shown in Fig. 3. Finally, the sender sends $\{\hat{E}_{0,i}, \hat{E}_{1,i}\}$ to the receiver.

b) *Data encryption:* Before the sender runs the secret sharing algorithm $SS()$, the sender chooses $2t$ random numbers $\{\Gamma_{0,i} \in_R G_q, \Gamma_{1,i} \in_R G_q\}_{i=1}^{t}$ as one-time pads ($\Gamma_{0,i}$ and $\Gamma_{1,i}$ should be temporarily stored and can be deleted at the end of the phase). Then, she randomizes $\{\hat{z}_{0,i}, \hat{z}_{1,i}\}_{i=1}^{t}$ as $u_{0,i} = \hat{z}_{0,i} \cdot \Gamma_{0,i}$ and $u_{1,i} = \hat{z}_{1,i} \cdot \Gamma_{1,i}$ for $i = 1$ to t. Then, the inputs of the secret sharing algorithm $SS()$ are changed from $z_{0,i}$ to $u_{0,i}$ and from $z_{1,i}$ to $u_{1,i}$.

c) *Data decryption:* The receiver encrypts $\{u_{b_i,i}\}_{i=1}^{t}$ as $\{u'_{b_i,i} = \mathsf{Enc}'_{\mathrm{rek}}(u_{b_i,i})\}_{i=1}^{t}$, and sends $\{u'_{b_i,i}\}_{i=1}^{t}$ to the sender. The sender encrypts $\{\Gamma_{0,i}, \Gamma_{1,i}\}_{i=1}^{t}$ as $\{\Gamma'_{0,i} = \mathsf{Enc}'_{rek}((\Gamma_{0,i})^{-1}), \Gamma'_{1,i} = \mathsf{Enc}'_{\mathrm{rek}}((\Gamma_{1,i})^{-1})\}_{i=1}^{t}$, and combines these two ciphertexts by additively homomorphic operations as $\bar{u}_{0,i} = u'_{b_i,i} \cdot \Gamma'_{0,i}$ and $\bar{u}_{1,i} = u'_{b_i,i} \cdot \Gamma'_{1,i}$, for $i = 1$ to t. She, then, reversely permutes $\{\bar{u}_{0,i}, \bar{u}_{1,i}\}$ to $\{w_{0,i}, w_{1,i}\}$ (reversed column permutation π_2^{-1}), and forwards the results to the receiver. The receiver decrypts $\{w_{0,i}, w_{1,i}\}$ as $\{z'_{0,i} = \mathsf{Dec}'_{\mathrm{rdk}}(w_{0,i}), z'_{1,i} = \mathsf{Dec}'_{\mathrm{rdk}}(w_{1,i})\}_{i=1}^{t}$ to obtain $z'_{0,i}$ and $z'_{1,i}$. The receiver decrypts the encrypted data blocks as $M_{b_i,i} = \mathsf{Dec}_{\mathsf{H}_1(z'_{b_i,i})}(\hat{E}_{b_i,i})$ for $i = 1$ to t.

3) *Key Tracing:* Assuming that the receiver leaks t' data blocks, the number of valid bits leaked by the receiver is κ', and the probability that the sender can successfully trace the data leaker and guess his/her private key rsk is $\mathsf{Pr}(\mathrm{Succ}) = (\frac{1}{2})^{\kappa-\kappa'}$. We can utilize the method of indicator random variables to calculate the expectation of κ', $E(\kappa')$.

For $i = 1$ to κ, let $I_i = 1$, if the bit b_i is leaked at least once, and let $I_i = 0$, otherwise. Then, the number of valid bits leaked is $\kappa' = \sum_{i=1}^{\kappa} I_i$, and by the linearity of expectation, we can calculate the expected number of valid bits leaked as $E(\kappa') = \sum_{i=1}^{\kappa} E(I_i)$. Specifically, $E(I_i)$ can be computed as

$$\begin{aligned} E(I_i) &= 1 \cdot \mathsf{Pr}(I_i = 1) + 0 \cdot \mathsf{Pr}(I_i = 0) \\ &= 1 \cdot \mathsf{Pr}(I_i = 1) = 1 - \mathsf{Pr}(I_i = 0). \end{aligned}$$

In R-ADS, each bit b_i is embedded ϵ times into t data blocks, and thus, the number of data blocks that do not involve the bit b_i is $t - \epsilon$. If the receiver randomly chooses t' out of t data blocks, the number of possible arrangements (of the chosen data blocks) is $\binom{t}{t'} = \frac{t!}{(t-t')!t'!}$. When there exists one restriction that the chosen data blocks do not involve the bit b_i, the number of possible arrangements is $\binom{t-\epsilon}{t'} = \frac{(t-\epsilon)!}{(t-\epsilon-t')!t'!}$. Then, we have $\mathsf{Pr}(I_i = 0) = \frac{\binom{t-\epsilon}{t'}}{\binom{t}{t'}}$. Finally, the expected number of leaked valid bits is calculated as follows:

$$E(\kappa') = \sum_{i=1}^{\kappa} (1 - \mathsf{Pr}(I_i = 0)) = \kappa \cdot \left(1 - \frac{\binom{t-\epsilon}{t'}}{\binom{t}{t'}}\right).$$

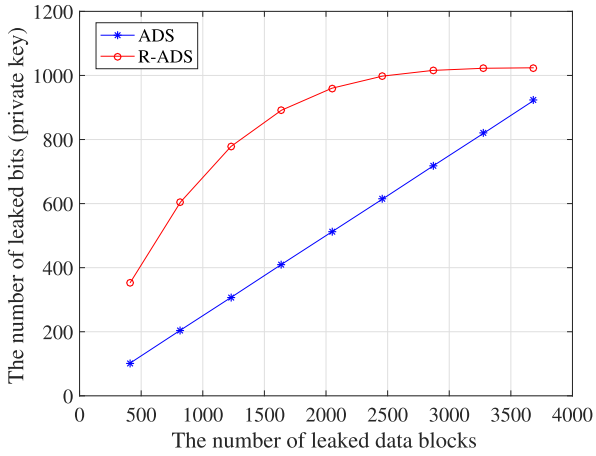


Fig. 4. Effectiveness of key tracing ($\kappa = 1024$, $\epsilon = 4$).

R-ADS mitigates the special attack mentioned in ADS since the receiver is prevented from learning the correspondence between the choices (bits b_i) and the watermarked data blocks $M_{b_i,i}$ for $i = 1$ to t. When the receiver chooses to leak the shared data, the leakage of his/her private key is random. Fig. 4 is utilized to demonstrate the effectiveness of key tracing in R-ADS compared with ADS. When the number of leaked data blocks is the same in ADS and R-ADS, the number of leaked bits (of private key) in R-ADS is more. For example, when the number of leaked data blocks is 2000 (among 4096 data blocks), the number of leaked bits in ADS is around 500 and the number of leaked bits in R-ADS is around 900.

D. Details of E-ADS

ADS and R-ADS are secure under the semihonest adversary model, but not secure under the malicious adversary model. If some SPs are malicious and perform arbitrary operations such as giving arbitrary values to the receiver as the shares, the receiver cannot detect the malicious SPs and verify that the received data are correct. Therefore, we introduce E-ADS, by utilizing the zero-knowledge proof [15] to verify the operations of SPs. Similarly, we only emphasize the differences between R-ADS and E-ADS in the following details, and ignore the repetitive parts.

1) *System Setup:* Another distinct generator $\mathfrak{g}$ is chosen from G_q and is published as the public parameter.

2) *Data Sharing:* The subphases, including *data preparation*, *data encryption*, and *data transfer*, are different from the corresponding subphases in R-ADS.

a) *Dta preparation:* The sender chooses $2t$ random numbers $\{\rho_{0,i} \in_R Z_q^*, \rho_{1,i} \in_R Z_q^*\}_{i=1}^{t}$, and calculates $2t$ group elements as $\{z_{0,i} = \mathfrak{g}^{\rho_{0,i}}, z_{1,i} = \mathfrak{g}^{\rho_{1,i}}\}_{i=1}^{t}$.

b) *Data encryption:* The sender sets $\Gamma_{0,i} = \mathfrak{g}^{\lambda_{0,i}}$ and $\Gamma_{1,i} = \mathfrak{g}^{\lambda_{1,i}}$, where $\lambda_{0,i} \in_R Z_q^*$ and $\lambda_{1,i} \in_R Z_q^*$. The inputs of the secret sharing algorithm $SS()$ are $\{\rho_{0,i} + \lambda_{0,i}, \rho_{1,i} + \lambda_{1,i}\}_{i=1}^{t}$, and the secret sharing function is changed to $f(x) \bmod q$. The sender also

calculates $\{g^{\rho_{0,i}+\lambda_{0,i}}, g^{\rho_{1,i}+\lambda_{1,i}}\}_{i=1}^{t}$ and for secret sharing functions $f(x)_{0,i}$ and $f(x)_{1,i}$ ($i = 1$ to t), the sender calculates $\{g^{a_{1,l,i}}, \ldots, g^{a_{k-1,l,i}}\}_{l=\{0,1\}}$. $\{g^{\rho_{0,i}+\lambda_{0,i}}, g^{\rho_{1,i}+\lambda_{1,i}}, \{g^{a_{1,l,i}}, \ldots, g^{a_{k-1,l,i}}\}_{l=\{0,1\}}\}_{i=1}^{t}$ are broadcasted to n SPs. Each SP calculates $g^{f(j)_{0,i}}$ and $g^{f(j)_{1,i}}$ as follows, and broadcasts the results to n SPs, such that n SPs have the same $\{g^{f(j)_{0,i}}, g^{f(j)_{1,i}}\}_{i=1,j=1}^{i=t,j=n}$

$$g^{f(j)_{0,i}} = g^{\rho_{0,i}+\lambda_{0,i}} \prod_{\tau=1}^{k-1} (g^{a_{0,\tau,i}})^{j^l}$$

$$g^{f(j)_{1,i}} = g^{\rho_{1,i}+\lambda_{1,i}} \prod_{\tau=1}^{k-1} (g^{a_{1,\tau,i}})^{j^l}.$$

If any dispute happens, n SPs use the majority voting method to reach a consensus.

c) *Data transfer:* The jth SP sends the shares $\{c_{0,i,j}, c_{0,i,j}\}_{i=1}^{t}$ to the receiver

$$c_{0,i,j} = \{g^{\delta_{0,i,j}}, \mathfrak{g}^{f(j)_{0,i}} \cdot (y_i/h^0)^{\delta_{0,i,j}}\}$$

$$c_{1,i,j} = \{g^{\delta_{1,i,j}}, \mathfrak{g}^{f(j)_{1,i}} \cdot (y_i/h^1)^{\delta_{1,i,j}}\}.$$

Simultaneously, it attaches a noninteractive zero-knowledge proof as follows, and other SPs verify the proof. If a majority of the SPs acknowledge the failure of the proof, the corresponding SP is identified as the malicious SP

$$\text{ZkPoK}\{(\{f(j)_{0,i}, f(j)_{1,i}, \delta_{0,i,j}, \delta_{1,i,j}\}_{i=1}^{t})$$
$$c_{0,i,j} = \{g^{\delta_{0,i,j}}, \mathfrak{g}^{f(j)_{0,i}} \cdot (y_i/h^0)^{\delta_{0,i,j}}\}$$
$$\wedge\, c_{1,i,j} = \{g^{\delta_{1,i,j}}, \mathfrak{g}^{f(j)_{1,i}} \cdot (y_i/h^1)^{1,i,j}\}$$
$$\wedge\, A_{0,i,j} = g^{f(j)_{0,i}} \wedge A_{1,i,j} = g^{f(j)_{1,i}}\}.$$

When performing the reconstruction algorithm $RC()$, line 6 changes to $w_{b_i,i} = \prod_{j=1}^{k} (v_{b_i,i,j})^{\prod_{l=0,l\neq i}^{k} \frac{l}{l-i}}$.

To reduce the computational cost at the sender side and the receiver side, the exponential operations and public-key operations can be outsourced to k SPs under the semihonest setting and the malicious setting. The sender and receiver can split the secrets (e.g., the exponents in the exponential operations) to k SPs and aggregate the results later. For the malicious setting, the results of these computations should be publicly verified. To achieve the goal, the prover can utilize the noninteractive zero-knowledge proof for composite statements [15], i.e., the secret is hashed first and then published later. For instance, to prove $\Theta = g^{\theta}$ while keeping $\theta \in Z_q^*$ secret, the prover can publish θ' and perform the following noninteractive zero-knowledge proof

$$\text{ZkPoK}\{(\theta) : \theta' = \mathsf{H}_1(\theta) \wedge \Theta = g^{\theta}\}.$$

IV. Security Analysis

In this section, we analyze how ADS, R-ADS, and E-ADS achieve the following three security requirements: data confidentiality, data accountability, and data correctness.

1) *Data Confidentiality ($\mathcal{A}_1$ and $\mathcal{A}_2$):* ADS and its variants allow up to $k-1$ SPs being corrupted and allow these corrupted SPs to collude with the sender or the receiver to break the data confidentiality.

Claim 1: Up to $k-1$ out of n collusive SPs cannot obtain the shared data of the sender $M_{0,i}$ and $M_{1,i}$ for $i = 1$ to t.

Analysis: $M_{0,i}$ and $M_{1,i}$ are encrypted using the symmetric key cryptosystem and the secret keys $z_{0,i}$ and $z_{1,i}$. As long as the symmetric key cryptosystem is semantically secure, the malicious SP cannot recover $M_{0,i}$ and $M_{1,i}$ from $E_{0,i}$ and $E_{1,i}$. In the meantime, $z_{0,i}$ and $z_{1,i}$ are shared with n SPs based on the Shamir's secret sharing scheme. Hence, unless there exist more than $k-1$ collusive SPs $z_{0,i}$ and $z_{1,i}$ cannot be recovered, and $M_{0,i}$ and $M_{1,i}$ cannot be obtained by malicious SPs.

Claim 2: The choice b_i (corresponding to a bit of private key rsk) of the receiver is unconditionally secure unless the receiver leaks the shared data blocks.

Analysis: In ADS, for each bit b_i, there is a randomness $r_i \in_R Z_q^*$ that satisfies $y_i = g^{r_i} h^{b_i}$. Therefore, the sender and SPs cannot get any information about the receiver's private key even if they have unlimited computational power. In R-ADS and E-ADS, since the zero-knowledge proof ZkPoK satisfies the *zero-knowledge* property, r_i and b_i are not exposed. Although the sender also receives the ciphertext $u'_{b_i,i}$ of the shared data, which is encrypted by the public key rek of the receiver, he cannot decrypt the ciphertext to obtain the shared data $u_{b_i,i}$ and cannot identify b_i without the private key of the receiver rdk, considering that the homomorphic encryption, e.g., Elgamal encryption, is semantically secure under the DDH assumption.

Claim 3: The receiver, even if colluding with up to $k-1$ out of n SPs, cannot extract $z_{1-b_i,i}$, and, thus, cannot obtain $M_{1-b_i,i}$.

Analysis: The jth SP creates the share of $z_{1-b_i,i}$ as $c_{1-b_i,i,j} = \{g^{\delta_{1-b_i,i,j}}, f(j)_{1-b_i,i} \cdot (y_i/h^{1-b_i})^{\delta_{1-b_i,i,j}}\}$. If there exists a probabilistic polynomial-time adversary $\mathcal{A}$ that can distinguish $(g, h, g^{\delta_{1-b_i,i,j}}, f(j)_{1-b_i,i} \cdot (y_i/h^{1-b_i})^{\delta_{1-b_i,i,j}})$ and (e_1, e_2, e_3, e_4) where $e_1, e_2, e_3, e_4 \in_R G_q^4$, $\mathcal{A}$ can also solve the DDH problem. The security proof is identical to the proof in [14]. In other words, $c_{1-b_i,i,j}$ does not reveal any useful information about $z_{1-b_i,i}$. Therefore, the receiver cannot extract $z_{1-b_i,i}$.

2) *Data Accountability ($\mathcal{A}_1$ and $\mathcal{A}_2$):* To ensure the data accountability, ADS and its variants apply three main mechanisms: the watermarking techniques [10], [13], the zero-knowledge proof technique [15], and the posterior investigation mechanism [11].

Claim 4: When the shared data are leaked, the sender can trace the responsibility of the receiver.

Analysis: The zero-knowledge proof technique guarantees that the receiver has to embed his/her privacy key rsk (the choice b_i) into the shared data M before receiving the shared data. Since the zero-knowledge proof ZkPoK

is *complete*, the receiver can easily generate the correct proof π_{key} when the receiver has the private key rsk. Since the zero-knowledge proof ZkPoK is *sound*, the receiver cannot deceive the sender and SPs to forge a valid private key rsk and obtain the shared data M_i. Particularly, ADS requires the receiver to make a commitment of each choice b_i as $y_i = g^{r_i} h^{b_i}$ and prove that the bits $\{b_i\}_{i=1}^{t}$ can be used for recovering the private key rsk. Moreover, the private key is embedded ϵ times, i.e., $t = \epsilon \times \kappa$, to increase the success possibility of key tracing. To prevent the attack where the receiver just leaks partial shared data, the permutation operations π_1 and π_2 are exploited such that the receiver cannot match the choice b_i with the received data block $M_{b_i,i}$. In the key tracing phase, the sender can obtain more bits of the private key rsk within this setting. Therefore, the sender can trace the responsibility of the receiver by extracting the private key according to the leaked shared data blocks and by applying a posterior investigation mechanism.

3) *Data Correctness ($\mathcal{A}_2$):* To ensure the data correctness, E-ADS applies the zero-knowledge proof of composite statements.

Claim 5: Up to $k-1$ SPs corrupt and behave maliciously, these malicious SPs can be detected.

Analysis: First, the shares of each SP are verified and the public verification is achieved via the zero-knowledge proof. The jth SP's ith shares $\{\mathfrak{g}^{f(j)_{l,i}}\}_{l=\{0,1\}}$ are proved that they are not manipulated based on the noninteractive zero-knowledge proof. The proof is constructed on the Σ-protocol in the random oracle model. This classical noninteractive zero-knowledge proof is secure based on Fiat–Shamir heuristic if the hash function is simulated as the random oracle [17]. That is, any misbehavior can be detected. Hence, the receiver can obtain the correct shares to recover $\{u_{b_i,i}\}_{i=1}^{t}$ even if some SPs are malicious. Second, the computations, performed by SPs, are also proved and verified based on the noninteractive zero-knowledge proof. When the proof is only related to the algebraic statement, the proof is also constructed on the Σ-protocol in the random oracle model. When the proof is related to the arithmetic statement, the proof is constructed on the zero-knowledge succinct noninteractive argument of knowledge. The security of composite statement has been discussed in [15].

V. Performance Evaluation

In this section, we evaluate the performance of ADS. To demonstrate the advantages of ADS, we compare ADS with two existing schemes, OT+ZkPoK [16] and Mangipudi *et al.* scheme [12] in the experiment. OT+ZkPoK is a straightforward combination of the simplest OT protocol and the zero-knowledge proof technique. Mangipudi *et al.* [12] scheme achieves the similar function as ADS. As the computational efficiency is one of the most important performance metrics in data sharing, the performance evaluation mainly focuses on the computational overhead and delay of the sender and the receiver in the phase of data sharing.

TABLE I
COMPUTATIONAL COMPLEXITY (SENDER)

Schemes	SKE	$HASH$	EXP	MUL	PKE
OT+ZkPoK [16]	$4t$	$8t$	$5t+3$	$t+1$	N/A
Mangipudi et al. [12]	$4t$	$8t$	$6t+3$	$2t+1$	$3t$
ADS	$2t$	$2t$	N/A	N/A	N/A
R-ADS	$2t$	$2t$	N/A	$4kt+2t$	N/A
E-ADS	$2t$	$6t$	N/A	$(k^2+6k+1)t$	N/A

TABLE II
COMPUTATIONAL COMPLEXITY (RECEIVER)

Schemes	SKE	$HASH$	EXP	MUL	PKE
OT+ZkPoK [16]	$3t$	$4t$	$2t$	t	N/A
Mangipudi et al. [12]	$3t$	$4t$	$5t$	$4t$	t
ADS	$2t$	$2t+1$	N/A	$(k^2+k)t+2k$	N/A
R-ADS	$2t$	$2t+1$	N/A	$(k^2+5k-1)t+2k$	N/A
E-ADS	$2t$	$(k+7)t+3$	N/A	$(k^2+5k-1)t+2k$	N/A

A. Computational Complexity

We first analyze ADS, R-ADS, and E-ADS in terms of the computational complexity of the sender and the receiver. We count the number of the time-consuming operations, including the symmetric-key encryption/decryption SKE, the cryptographic hash function HASH, the exponential operation EXP in G_q, the multiplicative operation MUL in G_q, and the public-key encryption/decryption PKE. The comparison results are shown in Tables I and II. Compared with the existing schemes [12], [16], ADS does not require any exponential operation or public-key encryption/decryption. These operations can be transformed to multiplicative operations to reduce the computational costs. An exponential operation at the sender side or the receiver side can be calculated by k SPs, and the sender and the receiver can combine the received results from k SPs by multiplication to obtain the final result, i.e., 1 EXP can be transformed to $k-1$ MULs. Similarly, the public-key encryption/decryption can be transformed as well. In R-ADS and E-ADS, the sender and the receiver have to compute more hash functions and do more multiplication. There are three reasons as follows:

1) the sender and the receiver need to perform public-key encryption, permutation, and randomization operations;
2) the sender needs to make a commitment to the secret sharing functions $f(x)$ for the later verification, and needs to do some exponentiation;
3) the sender and the receiver need to verify the results calculated by distributed k SPs, and an additional computation of a hash function is necessary for each verification.

B. Experiment Settings and Results

Table III shows the experiment setting. We choose SHA-256 algorithm as the cryptographic hash function and select the AES algorithm as the symmetric encryption/decryption algorithm. The Elgamal cryptosystem is selected as the public-key homomorphic encryption/decryption algorithm. The security parameter κ is set as 1024, the embedded times of the private key ϵ are set as $\{1, 2, 3, 4\}$, corresponding to the number of data

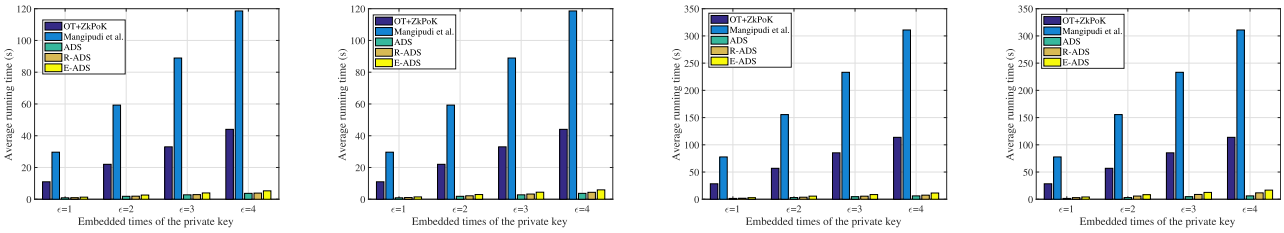


Fig. 5. Computational delay of the sender using the desktop and the Android smartphone. (a) Desktop (threshold $k = 2$). (b) Desktop (threshold $k = 10$). (c) Smartphone (threshold $k = 2$). (d) Smartphone (threshold $k = 10$).

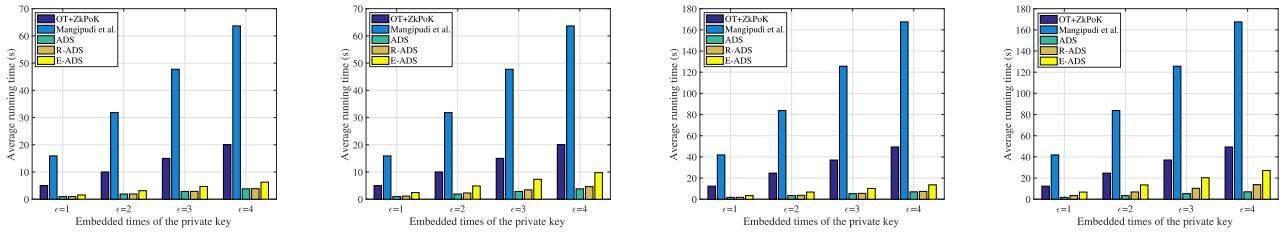


Fig. 6. Computational delay of the receiver simulated using the desktop and the Android smartphone. (a) Desktop (threshold $k = 2$). (b) Desktop (threshold $k = 10$). (c) Smartphone (threshold $k = 2$). (d) Smartphone (threshold $k = 10$).

TABLE III
EXPERIMENT SETTING

Parameters	Settings
Hash Function	SHA-256
Public Key Encryption	Elgamal Cryptosystem
Symmetric Key Length η	128
Symmetric Key Encryption	AES Algorithm
Security parameter κ	1024
Embedded Times ϵ	$1, 2, 3, 4$
Data Blocks Number t	$1024, 2048, 3072, 4096$
Size of Each Data Block	1 KB
Data Size	1 MB, 2 MB, 3 MB, 4 MB
Threshold (k, n)	$(2, 3)$ and $(10, 19)$

blocks as $\{1024, 2048, 3072, 4096\}$. The threshold k is set as $\{2, 10\}$, and each data block transmitted is 1 KB. Two devices are chosen as the sender or the receiver: a desktop (Windows 10 with 3.4 GHz processor and 32 GB memory) and a smartphone (Android 9.0 with 2.6 GHz processor and 6 GB memory).

We build a simulator for OT+ZkPoK [16], Mangipudi *et al.* scheme [12], ADS, R-ADS, and E-ADS based on JAVA language. The simulator simulates all computational operations of the sender and the receiver in the data sharing phase. We run the simulator 100 times and obtain the average computational delays in Fig. 5 and Fig. 6. As shown in these figure, ADS is more computationally efficient than OT+ZkPoK and Mangipudi *et al.* [12] scheme since the computational delay of the sender and the receiver is much smaller. The reason is that the sender and the receiver in ADS do not need to perform exponential operations and public-key encryption/decryptio operations. These operations are more time consuming than multiplicative operations. Also, with the increase of the private key's embedded times ϵ, the computational latency of the sender and the receiver increases as well. The reason is that, once the embedded times ϵ increase from 1 to 4, the rounds of OT protocol increase linearly, which leads to computational operations rise linearly. Moreover, to study the influence of the threshold k, different values for k are set in the simulation. The simulation results demonstrate that k does not significantly affect the computational efficiency of the sender and the receiver, i.e., the computational delay just increases a little. The reason is that, the number of the data blocks t is too large to cover the influence of k. In addition, compared with the existing schemes, ADS can run at the smartphone with acceptable delay (almost 30 s when $k = 10$) while the existing schemes takes more than 2 m to accomplish the computations when $\epsilon = 4$. Furthermore, both the sender and the receiver in R-ADS and E-ADS, compared with ADS, require more computational costs in the data sharing phase. The reason is that, they need to do more multiplicative operations and perform more hash functions to support the verifiable property and achieve the requirement of data correctness. Considering that these operations are not time-consuming compared with the exponential operations, the computational delay is also acceptable.

VI. RELATED WORKS

The security and privacy issues of data sharing have been well studied, and most of which focus on achieving fine-grained data access control [18]–[23] and privacy preservations [24]–[26]. There exist some papers [27]–[32] that discuss the data accountability issue, but these proposed schemes address the issue from different perspectives other than ADS. In addition, a bunch of papers about traitor tracing [33]–[36], is also related to

the accountability issue for data sharing, i.e., tracking the source of data breaches.

A. Accountability in Data Sharing

To address the accountability issue for data sharing, Guo *et al.* [27] introduced the concept of accountable proxy re-encryption (APRE) and proposed an APRE-based scheme, which enables anyone to judge whether the proxy (cloud) abuses the re-encryption key and re-encrypts a client's data without authorization. The accountability defined in their scheme is to trace the responsibility of the proxy (cloud). Li *et al.* [28] proposed a fine-grained data sharing scheme based on the attribute-based encryption. The identities of adversaries who publish their decryption boxes can be traced since their identities need to be embedded into the decryption box and can be publicly extracted. The accountability defined in their scheme is to trace the responsibility of the entity who has decryption capability. Different from them, Cao *et al.* [29] proposed a trust-based model for data sharing, where trust policies are deployed to each entity. The data usage history is recorded in a trusted authority such that anyone can trace the responsibility of data abusing. To get rid of the trusted authority, Neisse *et al.* [32] proposed a blockchain-based data accountability and provenance tracking scheme, where all logs of data processing are recorded on a decentralized blockchain. Similar to [32], Shahriar *et al.* [30] proposed an accountable data sharing scheme using blockchain. Their scheme relies on a relaxed trust assumption that only the data providers are trusted. The data accountability issues can be addressed by misbehavior report and automated punishment. Liang *et al.* [31] also proposed a decentralized accountable system for healthcare data based on blockchain. They applied a security hardware, Intel SGX, to guarantee the trustworthiness of data access logs, since all data access operations should be verified by the hardware, which is difficult to be compromised. Different from ADS, although these schemes also aim to achieve the accountability, they do not consider the case where the shared data are leaked and how to trace the adversary based on the leaked data, and cannot be directly applied in IIoT scenarios.

B. Traitor Tracing

The concept of traitor tracing is proposed by Chor *et al.* [33]. They called a receiver traitor, who allows nonauthorized entities to obtain the shared data from the sender. Particularly, the sender can encrypt the shared data using one master secret key and publish the encrypted shared data to a public platform. When the sender wants to share the data with a receiver, she can generate a decryption box with a unique private key and share the decryption box with the receiver. Then, the receiver can decrypt the encrypted shared data using the decryption box and the private key to obtain the data. Obviously, a traitor can make copies of her decryption box and private key to resell or make them public, thus allowing illegitimate entities to decrypt the encrypted shared data. Though this behavior cannot be prevented, it can be resisted in a traitor tracing scheme. A traitor tracing scheme can achieve the tracing goal even if a traitor modifies his/her private key before releasing it or a coalition of traitors work together to create a new decryption box and a new private key. Recently, the traitor tracing mechanisms are fully developed. Kiayias *et al.* [36] combined the blockchain technology and the traditional traitor tracing mechanism to propose a traitor deterring scheme. In their scheme, a traitor provides collateral for retrieving the unique private key used for decryption and the security of the collateral is protected through a smart contract, i.e., it is not revealed unless the traitor misbehaves. Nishimaki *et al.* [34] proposed an anonymous traitor tracing scheme where the identity information about the traitor is privacy preserving and is embedded into the traitor's private key and can only be recovered by the tracing algorithm. Goyal *et al.* [35] proposed the first collision-resistant traitor tracing scheme that is proven to be secure under the standard assumption. Although these traitor tracing schemes are effective to some extent, they are not fit for the case of data sharing in IIoT since ADS aims to support data leakage detection rather than key leakage detection. Also, ADS guarantees that the sender cannot extract the secret of the receiver unless the receiver leaks the shared data, but these traitor tracing schemes cannot satisfy the requirement.

VII. Conclusion

In this article, we proposed ADS for IIoT. ADS enabled a data owner to share any data with a data receiver, but prevents the latter from exposing the shared data to the public without permission. If the shared data are leaked, the data owner can extract the data leaker's private key from the leaked data to pursue his/her responsibility. Even if the data leaker only leaks part of the shared data, the sender still has a large probability to recover his/her private key. In addition, compared with the existing schemes, ADS significantly reduced the computational overhead of the sender and the receiver while the security can still be guaranteed, which makes ADS well fit for IIoT. In the future work, we aim to propose a blockchain-based accountable data sharing scheme without off-chain protocols, which does not only have low computational overhead but also low communication costs.

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[2] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3690–3700, Aug. 2018.

[3] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "Healthdep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4101–4112, Sep. 2018.

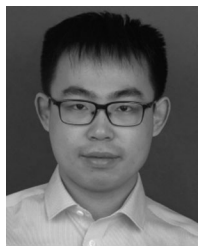
[4] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Providing task allocation and secure deduplication for mobile crowdsensing via fog computing," *IEEE Trans. Dependable Secure Comput.*, to be publhised, doi: 10.1109/TDSC.2018.2791432.

[5] C. H. Liu, Q. Lin, and S. Wen, "Blockchain-enabled data collection and sharing for industrial IoT with deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3516–3526, Jun. 2019.

[6] J. Kang *et al.*, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4660–4670, Jun. 2019.

[7] F. Lyu *et al.*, "Intelligent context-aware communication paradigm design for IoVs based on data analytics," *IEEE Netw.*, vol. 32, no. 6, pp. 74–82, Nov./Dec. 2018.

[8] J. Liang, Z. Qin, S. Xiao, L. Ou, and X. Lin, "Efficient and secure decision tree classification for cloud-assisted online diagnosis services," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2922958.
[9] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. Image Process.*, vol. 6, no. 12, pp. 1673–1687, Dec. 1997.
[10] E. Ayday, E. Yilmaz, and A. Yilmaz, "Robust optimization-based watermarking scheme for sequential data," in *Proc. USENIX RAID*, 2019, pp. 323–336.
[11] V. Buterin, "A next-generation smart contract and decentralized application platformm," 2019, [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper, Accessed on: Jun. 3, 2019.
[12] E. V. Mangipudi, K. Rao, J. Clark, and A. Kate, "Towards automatically penalizing multimedia breaches," Cryptology ePrint Archive, Report 2018/1050, 2018. [Online]. Available: https://eprint.iacr.org/2018/1050
[13] D. Boneh, A. Kiayias, and H. W. Montgomery, "Robust fingerprinting codes: A near optimal construction," in *Proc. ACM Digit. Rights Manage.*, 2010, pp. 3–12.
[14] W. Tzeng, "Efficient 1-out-of-n oblivious transfer schemes with universally usable parameters," *IEEE Trans. Comput.*, vol. 53, no. 2, pp. 232–240, Feb. 2004.
[15] S. Agrawal, C. Ganesh, and P. Mohassel, "Non-interactive zero-knowledge proofs for composite statements," in *Proc. CRYPTO*, 2018, pp. 643–673.
[16] T. Chou and C. Orlandi, "The simplest protocol for oblivious transfer," in *Proc. LATINCRYPT*, K. E. Lauter and F. Rodríguez-Henríquez, Eds., 2015, vol. 9230, pp. 40–58.
[17] C. Huang, R. Lu, X. Lin, and X. Shen, "Secure automated valet parking: A privacy-preserving reservation scheme for autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 169–11 180, Nov. 2018.
[18] S. Xu, G. Yang, Y. Mu, and R. H. Deng, "Secure fine-grained access control and data sharing for dynamic groups in the cloud," *IEEE Trans. Inform. Forensics Secur.*, vol. 13, no. 8, pp. 2101–2113, Aug. 2018.
[19] Q. Huang, L. Wang, and Y. Yang, "DECENT: Secure and fine-grained data access control with policy updating for constrained IoT devices," *World Wide Web*, vol. 21, no. 1, pp. 151–167, 2018.
[20] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.
[21] J. Xiong *et al.*, "A personalized privacy protection framework for mobile crowdsensing in IIoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4231–4241, Jun. 2020.
[22] S. Xu, Y. Li, R. Deng, Y. Zhang, X. Luo, and X. Liu, "Lightweight and expressive fine-grained access control for healthcare Internet-of-Things," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2936481.
[23] J. Hao, J. Liu, W. Wu, F. Tang, and M. Xian, "Secure and fine-grained self-controlled outsourced data deletion in cloud-based IoT," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1140–1153, Feb. 2020.
[24] C. Xu, N. Wang, L. Zhu, K. Sharif, and C. Zhang, "Achieving searchable and privacy-preserving data sharing for cloud-assisted E-healthcare system," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8345–8356, Oct. 2019.
[25] S. Sharma, K. Chen, and A. P. Sheth, "Toward practical privacy-preserving analytics for iot and cloud-based healthcare systems," *IEEE Internet Comput.*, vol. 22, no. 2, pp. 42–51, Mar./Apr. 2018.
[26] J. Chen, J. He, L. Cai, and J. Pan, "Disclose more and risk less: Privacy preserving online social network data sharing," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2018.2861403.
[27] H. Guo, Z. Zhang, J. Xu, N. An, and X. Lan, "Accountable proxy re-encryption for secure data sharing," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2018.2877601.
[28] J. Li, X. Chen, S. S. M. Chow, Q. Huang, D. S. Wong, and Z. Liu, "Multi-authority fine-grained access control with accountability and its application in cloud," *J. Netw. Comput. Appl.*, vol. 112, pp. 89–96, 2018.
[29] Q. H. Cao, I. Khan, R. Farahbakhsh, G. Madhusudan, G. M. Lee, and N. Crespi, "A trust model for data sharing in smart cities," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–7.
[30] M. Shahriar Rahman, A. Al Omar, M. Z. A. Bhuiyan, A. Basu, S. Kiyomoto, and G. Wang, "Accountable cross-border data sharing using blockchain under relaxed trust assumption," *IEEE Trans. Eng. Manag.*, to be published, doi: 10.1109/TEM.2019.2960829.
[31] X. Liang, S. Shetty, J. Zhao, D. Bowden, D. Li, and J. Liu, "Towards decentralized accountability and self-sovereignty in healthcare systems," in *Proc. Int. Conf. Inf. Commun. Syst.*, 2017, vol. 10631, pp. 387–398.
[32] R. Neisse, G. Steri, and I. N. Fovino, "A blockchain-based approach for data accountability and provenance tracking," in *Proc. Int. Conf. Adv. Res. Eng. Sci.*, 2017, pp. 14:1–14:10.
[33] B. Chor, A. Fiat, M. Naor, and B. Pinkas, "Tracing traitors," *IEEE Trans. Inform. Theory*, vol. 46, no. 3, pp. 893–910, May 2000.
[34] R. Nishimaki, D. Wichs, and M. Zhandry, "Anonymous traitor tracing: How to embed arbitrary information in a key," in *Proc. EUROCRYPT*, 2016, pp. 388–419.
[35] R. Goyal, V. Koppula, and B. Waters, "Collusion resistant traitor tracing from learning with errors," in *Proc. ACM Symp. Theory Comput.*, 2018, pp. 660–670.
[36] A. Kiayias and Q. Tang, "Traitor deterring schemes: Using bitcoin as collateral for digital content," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2015, pp. 231–242.



Cheng Huang (Student Member, IEEE) received the B.Eng. and M.Eng. degrees in information security from Xidian University, Xi'an, China, in 2013 and 2016, respectively. He is currently working toward the Ph.D. degree in Electrical and Computer Engineering with the Department of Electrical and Computer Engineering, University of Waterloo, ON, Canada.

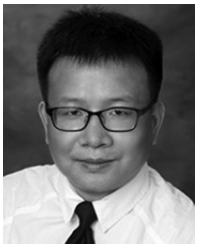
He was a Project Officer with the INFINITUS Laboratory, the School of Electrical and Electronic Engineering, Nanyang Technological University till July 2016. His research interest includes the areas of applied cryptography, cyber security and privacy in the mobile network.

Dongxiao Liu (Student Member, IEEE) received the B.S. and M.S. degrees in information security from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2013 and 2016, respectively. He is currently working toward the Ph.D. degree in Electrical and Computer Engineering with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada.

His research interests include applied cryptography and privacy enhancing technologies for blockchain.

Jianbing Ni (Member, IEEE) received the B.E. and M.S. degrees in information security from the University of Electronic Science and Technology of China, Chengdu, China, in 2011 and 2014, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2018.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON, Canada. His current research interests include applied cryptography and network security, with a focus on cloud computing, smart grid, mobile crowdsensing, and the Internet of Things.



Rongxing Lu (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2012.

He was an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, from 2013 to 2016. He has been an Associate Professor with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Fredericton, NB, Canada, since 2016. He was a Postdoctoral Fellow with the University of Waterloo, from 2012 to 2013. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy.

Dr. Lu was a recipient of the Governor Generals Gold Medal for his Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, the 8th IEEE Communications Society (ComSoc) AsiaPacific Outstanding Young Researcher Award in 2013, and the 2016 to 2017 Excellence in Teaching Award from FCS, UNB. He is currently with the ViceChair (Publication) of IEEE ComSoc CIS-TC.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular ad hoc and sensor networks.

Dr. Shen was the recipient of the R.A. Fessenden Award in 2019 from IEEE, Canada, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society. He has also received the Excellent Graduate Supervision Award in 2006 and Outstanding Performance Award five times from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He was the Technical Program Committee Chair/Co-Chair for the IEEE Globecom'16, the IEEE Infocom'14, the IEEE VTC'10 Fall, the IEEE Globecom'07, the Symposia Chair for the IEEE ICC'10, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He was the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL and IEEE NETWORK, and the Vice President on Publications of the IEEE Communications Society. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Fellow, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.