






# Blockchain-Based Fair and Fine-Grained Data Trading With Privacy Preservation

Liang Xue , *Member, IEEE*, Jianbing Ni , *Senior Member, IEEE*, Dongxiao Liu , *Member, IEEE*, Xiaodong Lin , *Fellow, IEEE*, and Xuemin Shen , *Fellow, IEEE*

**Abstract**—In this article, we propose a blockchain-based fair and privacy-preserving data trading scheme that supports fine-grained data selling. First, to achieve fairness for trading participants, by incorporating attribute-based credentials, encryption, and zero-knowledge proof, we design a data trading scheme where a buyer first publishes the required data attributes on the blockchain, and a data seller can demonstrate data availability in ciphertext by only disclosing the required attributes to a data buyer and proving the authenticity of data. A data buyer transfers funds only if the correct key material is uploaded to the blockchain. Second, to guarantee fine-grained data trading and preserve identity privacy, we build a Merkle hash tree on the ciphertexts of data with a signature on its root node, which allows a data seller to split data into blocks and remove the sensitive information from the data without affecting data availability verification. The public key of the data seller is not leaked to the data buyer during the trading. Moreover, different trading transactions from the same data seller cannot be linked. We formally prove that our scheme achieves the desired security properties: fairness and privacy preservation. Simulation results demonstrate the feasibility and efficiency of the proposed scheme.

**Index Terms**—Blockchain, fair data trading, fine-grained data selling, privacy preservation.

## I. INTRODUCTION

WITH the world going through the digital revolution, data have been one of the most valuable assets in our economy [1]. Data trading enables data buyers to purchase data from data sellers. Fair data trading requires that a data buyer can obtain the required data if it pays the corresponding funds, and a data seller can receive the funds if it sends the data to the data buyer and the latter obtains the data. However, in data trading, there is mistrust between a data seller and a data buyer, which would lead to a deadlock point where a data seller is reluctant to pay first until a data seller sends the data to it, and

the data seller would not provide its data until it receives the reward. The party who acts first would be at a disadvantage since the other party may be dishonest and disappear. In traditional data trading, there are trusted third parties, such as Dawex [2], Datacoup [3], and Xignite [4], which establish trust between the trading participants. Nevertheless, the trading platform may be dishonest and try to steal and resell data sellers' data. In addition, it can be a target of attackers and be vulnerable to the single point of failure attack.

The decentralized nature of blockchain makes it have the potential to achieve distributed data trading without a trusted third party [5]. Blockchain suits the role because of its transparency, programmability, and the embedded cryptocurrencies [6], [7]. Smart contracts on the blockchain can achieve automated payment when predefined conditions are satisfied. The process of blockchain-based data trading can be described as follows: A data buyer publishes the data requirements and deposits the reward of data on a smart contract. Then, a data seller demonstrates the availability of data to the data buyer. Considering that the size of data may be large and the storage of a smart contract is limited, a data seller can first encrypt the data with a key and store the ciphertexts at an off-chain storage platform, such as the InterPlanetary File System (IPFS) or a cloud server [8]. Once the data availability is verified, the data seller only needs to upload the corresponding key to the smart contract. If the key is correct, the reward is transferred to the data seller.

For blockchain-based fair data trading, there are two major challenging issues: 1) How to verify the data availability, i.e., data satisfy the requirements of a data buyer, without viewing the plaintext data. If the check is based on the plaintext, the data buyer can refuse to pay for the data even if the data are indeed what it needs as it has already owned the data; and 2) How to guarantee the data retrievability, i.e., the decryption key should be correct, and the data buyer can use the key to decrypt the ciphertext and obtain the data. Public predicate functions such as hash functions can be utilized to validate the availability of data [9], [10], [11], where encryption and complicated zero-knowledge proof are used to demonstrate that the data satisfy a predicate function. The method is not suitable for data that have no public verifiable predicate function, such as Internet of Things (IoT) data and email data. The authenticate-based method can also be used to verify the data availability by checking whether the data are signed by a trusted authority [12], and data retrievability is achieved by utilizing a re-encryption scheme. For this method, fine-grained data trading is not considered.

Manuscript received 26 August 2022; revised 21 December 2022; accepted 9 February 2023. Date of publication 2 March 2023; date of current version 9 August 2023. Recommended for acceptance by M. Correia. (Corresponding author: Dongxiao Liu.)

Liang Xue, Dongxiao Liu, and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: l34xue@uwaterloo.ca; dongxiao.liu@uwaterloo.ca; sshen@uwaterloo.ca).

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

Xiaodong Lin is with the School of Computer Science, University of Guelph, Guelph, ON N1G 2W1, Canada (e-mail: xlin08@uoguelph.ca).

Digital Object Identifier 10.1109/TC.2023.3251846

State-of-the-art blockchain-based fair data trading schemes mainly address the fairness issue in data trading, and few of them consider the privacy issues it might cause. For example, in healthcare data trading, an electronic medical record (EMR) contains much personal information, such as the name, age, and address of a patient. It is important to accomplish data trading without compromising the privacy of data sellers. For fair data trading, the data availability of an EMR can be verified by a signature signed by a hospital. With the original encrypted EMR and its signature, data authenticity is verified. However, it is a challenging task to prove data availability without sending the whole file to a data buyer, i.e., trading only part of a file while data availability can be verified. Moreover, the identity of a hospital can be used to infer the home address information of a patient. Thus, the identities of hospitals should be hidden in verifying data availability.

In this work, to address the above issues, we propose a blockchain-based fair and fine-grained data trading scheme with privacy preservation. We take healthcare data as an example. A data buyer wants to collect healthcare data from EMRs for further analysis. It can publish the data requirements for the data it needs, which include the acceptable hospitals that EMRs are issued from and data fields in EMRs that are required. For example, a data buyer needs data related to *Diabetes*, and data fields for *Blood Pressure*, *Insulin*, *Glucose* are required. For a data seller who has an EMR that satisfies the data requirements, it is desired that only the required data items are sent to the data buyer at the end. Other sensitive information, including the identities of the data seller and the hospital, is concealed from the data buyer. In our proposed scheme, for the demonstration of data availability, EMRs are signed by hospitals to guarantee the authenticity of data. By utilizing the attribute-based anonymous credential [13], [14], an authenticated data structure, and zero-knowledge proof, a data seller can prove that its EMR is signed by one of the acceptable hospitals and contains the data fields required by a data buyer, while other information is not exposed to the data buyer even after data trading. A data buyer cannot resell the data as the private key of the data seller is involved in proving the data availability. For data retrievability, an EMR is first encrypted with a symmetric key, which is then encrypted with the public key of a data seller. To enable a data buyer to decrypt the ciphertext of trading data, the data seller encrypts the symmetric key with the buyer's public key and uploads the ciphertext and a correctness proof of the ciphertext to the smart contract. With the public key of the data buyer and other auxiliary information, the proof can be verified and funds are transferred to the data seller by the smart contract. The contributions of the paper can be summarized as follows:

- We define the desired functionalities of fine-grained and fair data trading, which include fairness, privacy preservation, and fine-grained trading. We propose the first blockchain-based fair and fine-grained data trading scheme, where a data seller can sell data in part without affecting data availability verification. Privacy preservation for data sellers is achieved;
- To demonstrate the data availability, the data fields or attributes in an EMR are signed by the hospital that issues the EMR. The corresponding attribute values are encrypted

separately, and the order of the attributes and their values are ensured by an authenticated data structure. Zero-knowledge proof is used to preserve the privacy of data sellers and achieve fair data trading. Only the required data can be recovered by the data buyer, and other information in the EMR is concealed;

- We formulate a security model for blockchain-based fair and fine-grained data trading and formally prove that our scheme can achieve the desired properties. We simulate the proposed scheme, and simulation results show the practicability of our scheme. Also, the gas cost for the trading smart contract is low.

The remaining paper is organized as follows: In Section II, we review the related works. In Section III, we describe the system model, threat model, design goals, and security definitions. In Section IV, we introduce the building blocks for constructing our scheme. In Section V, we present the overview and the details of the constructed scheme. In Section VI, we show the security proof for the proposed scheme, followed by the performance evaluation in Section VII. Finally, we conclude the paper in Section VIII.

## II. RELATED WORKS

The problem of fair data trading has been studied for decades. With the emergence of blockchain-based cryptocurrencies, fair data trading can be achieved in a completely trustless manner, where blockchain replaces the role of the trusted party. As data availability and retrievability are challenging issues in the design of fair data trading scheme, in the following, we review the related works in terms of data availability and retrievability.

The first blockchain-based fair data trading scheme named ZKCP was proposed by Gregory Maxwell [15]. In the scheme, data availability can be verified by a public predicate function, such as a hash function. A data seller encrypts digital goods with a symmetric key, generates a zero-knowledge proof that proves the data encrypted satisfy a public predicate function. In ZKCP, data retrievability is achieved by revealing the encryption key of the data. ZKCP scheme incurs setup issues since the setup of the zero-knowledge proof system [16], which is a Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zkSNARK) system, is done by the data buyer. To address the issues in ZKCP, Li et al. proposed a fair-exchange scheme supporting practical data exchange that is called ZKPlus [11], in which a new commit-and-prove non-interactive zero-knowledge (CP-NIZK) argument of knowledge scheme is introduced to replace the zkSNARK in ZKCP. ZKPlus has a public setup as commitment schemes and can handle complicated data validation.

For the verification of data availability, besides the predicate function, a few research works use sampling techniques [17], [18], [19]. Delgado-Segura et al. presented a fair data trading scheme [17], where cut and choose method is used to prove data availability [17]. The data buyer can randomly choose a subset of data that are called samples and obtain the plaintext of the sample data. By verifying the correctness of the sample data, the buyer can validate the data availability. For data recoverability, a vulnerability of the elliptic curve digital signature algorithm (ECDSA) [20] is exploited to implicitly reveal the secret key that

TABLE I  
COMPARISON WITH RELATED WORKS

Scheme	Fairness	Privacy Preservation	Method for Data Availability	Method for Data Retrievability	Issuer Hiding	Fine-grained Data Selling
[15]	✓	✗	Predicate function	Key revealing	NA	✗
[11]	✓	✗	Predicate function	Key revealing	NA	✗
[17]	✓	✗	Sampling technique	Key recovery	NA	✗
[18]	✓	✗	Sampling technique	Key recovery	NA	✗
[7]	✓	✓	Committee member authentication	Key recovery	NA	✗
[12]	✓	✓	Signature authentication	Re-encryption key revealing	✗	✗
Our work	✓	✓	Signature authentication	Key recovery	✓	✓

can decrypt all the ciphertexts. Moreover, Zhao et al. proposed a machine learning-based fair data trading scheme [18], where a sampling technique and a distance metric learning method are used to prove the validity of data, and the double authentication preventing signature is used to recover the decryption key.

Another approach to verify the data availability is an authentication-based method. Liu et al. proposed a blockchain-cloud transparent data marketing scheme [7], where a distributed and trusted committee verifies the data availability and manages anonymous credentials for data sellers. Anonymous credentials allow a service provider to authenticate a user without leaking the user's identity. An attribute-based anonymous credential (ABAC) contains several attributes of a user that are authenticated by an authority, and it enables a user to hide certain attributes during authentication and maintain strong privacy. There has been much research on ABAC, for example, threshold-based ABAC [21], delegatable ABAC [22], and traceable ABAC [23].

Moreover, Galteland et al. proposed a blockchain-based privacy-preserving fair data trading scheme, where a data manager is introduced as a trusted authority [12]. Data signed by the manager are believed to be authentic. The manager encrypts data with the public key of the data owner and signs the data. The data buyer can verify the signature to attest to the data availability. The data owner then generates a re-encryption key, and it sends the key and a correctness proof of the key to a smart contract to prove data retrievability. With the re-encryption key, the data buyer can transform the original ciphertext into a ciphertext under its public key and decrypt it with its private key.

Islam et al. designed a scheme to achieve privacy-preserving news trading [24]. An authority called "block cop (BC)" is introduced to guarantee trading fairness. If a seller does not send the file after receiving the funds from a buyer, the buyer can report it to BC, which will investigate the fraud. In comparison, our scheme utilizes crypto techniques to ensure trading fairness and privacy preservation. Moreover, in [24], a data seller needs to generate multiple data ciphertexts for different buyers. In our scheme, only the data encryption key is encrypted by a buyer's public key.

In our work, we employ an authentication-based mechanism to prove the data availability. Instead of a public authority in the system, our scheme can support multiple issuers. Data signed by one of the issuers are acceptable and can be verified without leaking the identity of the issuer. Moreover, the proposed scheme can achieve fine-grained data selling by only exposing required data to data buyers, and the on-chain computational overhead of

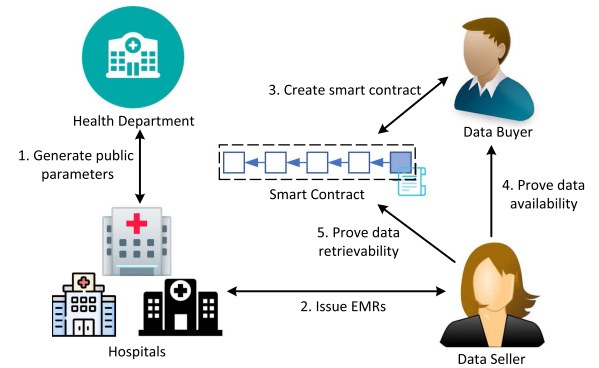


Fig. 1. System model.

our scheme is low. The comparison of related works is listed in Table I.

### III. SYSTEM MODEL

#### A. System Model

In our system model, there are five types of entities, a health department (HD), hospitals, data sellers, data buyers, and smart contracts, as shown in Fig. 1.

- *HD* – HD generates public system parameters for hospitals so that they can sign EMRs with the same public parameters. The hospitals are registered with HD. After that, HD publishes the public keys of hospitals;
- *Hospital* – A hospital can generate EMRs for patients and sign EMRs with its private key. Others can verify the authenticity of data by checking the signature of an EMR. For data trading, the identities of the hospitals are hidden from data buyers;
- *Data seller* – A data seller is a patient that owns EMRs and wants to monetize the data. During the trading, the seller is reluctant to leak its personal information to the data buyer.
- *Data buyer* – A data buyer is a collector of healthcare data, which can be government organizations, pharmaceutical industries, and healthcare professionals. A data buyer pays funds only if the data provided by a data seller satisfy its requirements.
- *Smart contract* – A smart contract is a self-executing contract deployed on a blockchain that runs if predetermined conditions are verified [25], [26]. When the conditions are met, the smart contract can automate the transfer of funds to guarantee the atomicity of a transaction.



There are two phases in our scheme: pre-trading and trading. In the pre-trading phase, HD generates the public parameters for the hospitals. Hospitals can issue and sign EMRs for patients. In the data trading phase, hospitals are not involved, which can reduce the burden of hospitals. A data buyer first creates a smart contract in which data requirements are defined, which are called policies in our scheme. The data buyer also deposits the rewards of data on the smart contract. If a data seller has EMRs that satisfy the defined policy, it can interact with the buyer and the smart contract to complete the trading.

### B. Threat Model

In our system, HD and hospitals are honest, and the smart contract is executed according to the defined logic. Data sellers and data buyers can be malicious, since they intend to obtain more benefits from the other party. A data seller who wishes to receive rewards may not have the required data. Thus, it would try to forge an EMR that can pass the verification of the data seller. A data buyer may refuse to pay the rewards when it obtains the desired data. Moreover, a data buyer intends to learn the real identity of a data seller. Other nodes in the blockchain network try to link different transactions of the same data seller and want to learn the identity information of the data seller.

### C. Desired Goals

The design goals of blockchain-based fair and fine-grained data trading are listed as follows:

- **Fairness** – For a data seller, fairness implies that after it sends the data satisfying the requirements of a data buyer and the latter obtains the data, it can receive the rewards. A data buyer cannot resell the data seller's data to other honest data buyers. If a data buyer pays the funds, it can obtain the data it needs;
- **Privacy preservation** – For a data seller, only the required data are disclosed to the data buyer. Sensitive information in an EMR is not obtained by the data buyer after data trading. A data seller should be able to prove data availability without leaking the personal information of the data seller and the hospital's identity. The smart contract does not reveal the identity of the data seller and transactions of the same data seller cannot be linked;
- **Fine-grained trading** – Data should be able to be traded in a fine-grained way. In data trading, a data seller can split data into many chunks and only send the data blocks that are required by a data buyer without affecting the verification of data availability.
- **Efficiency** – The off-chain computation, communication, and storage overhead for hospitals, data buyers, and data sellers should be small. Moreover, the execution cost or the gas fees consumed for the smart contract should be low.

### D. System Components

In the pre-trading phase, there are four algorithms, which are listed as follows:

- **Setup( $k$ )**: In this algorithm, HD takes as input a security parameter  $k$ , and outputs public parameters for hospitals;

- **Hospital Key Generation( $pp$ )**: This algorithm is run by a hospital. With inputs  $pp$ , the hospital generates a public-private signing key pair ( $hpk, hsk$ ) and registers itself with HD;
- **Issue( $hsk$ )**: In this algorithm, a hospital interacts with a patient to create an EMR for the patient. The patient generates its public-private key pair ( $upk, usk$ ), and the hospital generates an EMR for the patient based on the attributes and the patient's attribute values in the EMR;
- **Verify( $hpk, EMR$ )**: This algorithm is run by a patient. After receiving the EMR from the hospital, the patient verifies the validity of the EMR using  $hpk$ .

In the data trading phase, a data buyer interacts with a data seller, which is a patient in the pre-trading phase, and a smart contract to complete data trading. There are five algorithms in the data trading phase, which are listed below:

- **Policy Definition( $\{hpk\}_{i \in [1, \eta]}$ )**: This algorithm is run by a data buyer. The data buyer first generates its public-private key pair ( $bpk, bsk$ ). Then, given the acceptable hospitals whose public keys are  $\{hpk\}_{i \in [1, \eta]}$ , where  $\eta$  denotes the number of accepted hospitals, the data buyer signs  $\{hpk\}_{i \in [1, \eta]}$  and obtains the signatures ( $\{\sigma_i\}_{i \in [1, \eta]}$ ). It also defines the attribute requirements for the desired data, which specify the attributes that need to be included in an EMR, such as blood pressure and body mass index. Moreover, it can require the attribute value of a specific attribute. For example, the attribute value for diagnosis should be diabetes. The output of this algorithm is the defined policy ( $pol$ ). The data buyer creates a data trading smart contract  $SC$  and uploads  $pol$  to  $SC$ ;
- **Policy Verification( $pol, bpk$ )**: This algorithm is run by a data seller. Given  $pol$  and  $bpk$ , the data buyer verifies the validity of  $pol$ ;
- **Present( $EMR, pol, hpk$ )**: In this algorithm, a data seller generates a zero-knowledge proof proving that it has an EMR that satisfies  $pol$ . The output of the algorithm is a token  $pt$ . The data seller sends  $pt$  and auxiliary information, including the ciphertext of the trading data with a symmetric key ( $k$ ) being the secret encryption key, to the data buyer;
- **Present Verify( $pol, pt$ )**: This algorithm is run by a data seller. Given  $pol$ ,  $pt$ , and the auxiliary information, the data seller checks the data availability and data ownership of the data seller. If the requirements are satisfied, the data buyer uploads a confirmation message  $CM$  to  $SC$ ;
- **Ciphertext Generation( $CM, bpk$ )**: Given  $CM$  and  $bpk$ , the data seller generates a ciphertext ( $CB$ ) of  $k$  that can be decrypted by the data buyer's private key  $bsk$ . It also creates a zero-knowledge proof ( $\pi_b$ ) that proves the correctness of  $CB$ . The data seller then uploads  $\{CB, \pi_b\}$  to  $SC$ . After  $\{CB, \pi_b\}$  are verified by the validators of the blockchain,  $SC$  transfers the reward to the data seller.

### E. Security Definitions

Based on our design goal, a blockchain-based data trading scheme should satisfy two security properties: fairness and privacy preservation.

To guarantee the fairness to a data seller, it should satisfy that if a data seller sends the data to a data buyer and the buyer obtains the data, the data seller can receive the rewards. Thus, the atomicity of transactions should be satisfied. Moreover, a data buyer should not be able to resell a data seller's data, which implies that the data ownership should be verified. For the fairness to a data buyer, it should satisfy that if a data buyer transfers the funds, it should obtain the data it required instead of invalid data. Therefore, the data availability should be validated. The atomicity of transactions can be guaranteed by the automatic execution of smart contracts. For data ownership and data availability verification, it should be guaranteed that a data seller should be able to prove the ownership of an EMR, and it cannot forge an EMR with an acceptable hospital's signature, which is described as unforgeability.

Unforgeability requires that a data seller cannot forge a valid present token if it does not receive an EMR satisfying the requirements of a data buyer. The proposed scheme should be existential unforgeability under chosen message attack (EU-CMA). The secure game for unforgeability, where a challenger  $\mathcal{C}$  and an adversary  $\mathcal{A}$  are involved, is defined as follows.

*Definition 2.1 (GAME Unforgeability)*

*Setup.* Given a security parameter  $k$  and the number of hospitals  $n_H$  in the system, where  $n_H$  can be an arbitrary integer,  $\mathcal{C}$  runs Setup algorithm to obtain the public parameters  $pp$ .  $\mathcal{C}$  then runs Hospital Key Generation algorithm to generate the public-private key pairs  $(hpk_i, hsk_i)$  for  $i \in [1, n_H]$ .  $\mathcal{C}$  sets the set  $\mathcal{Q}_{issue}$  and the set  $\mathcal{Q}_{present}$  to be  $\emptyset$ . After that,  $\mathcal{C}$  sends  $pp$  and  $\{hpk_i\}_{i \in [1, n_H]}$  to  $\mathcal{A}$ .

*Issue Query.*  $\mathcal{A}$  can submit issue queries to  $\mathcal{C}$ . A  $j$ -th issue query can be denoted as  $(i_j, attr_{i_j}, \vec{D}_j)$ , where  $i_j$  represents a hospital with public key  $hpk_{i_j}$ , and  $\{attr_{i_j}, \vec{D}_j\}$  are attributes and attribute values in an EMR. After receiving an issue query,  $\mathcal{C}$  acts as an hospital who has  $hsk_{i_j}$  and generates an EMR for  $\mathcal{A}$ . Then,  $\mathcal{C}$  adds  $\{i_j, attr_{i_j}, \vec{D}_j\}$  to  $\mathcal{Q}_{issue}$  and sends the resulting EMR to  $\mathcal{A}$ .

*Present Query.*  $\mathcal{A}$  can also make present queries to  $\mathcal{C}$ , which can be denoted as  $(j, pol, EMR)$ . With the inputs  $(EMR, hpk_{i_j}, pol)$ ,  $\mathcal{C}$  runs Present algorithm, and sends the resulting  $pt$  and auxiliary information to  $\mathcal{A}$ . Then,  $\mathcal{C}$  adds  $\{j, pol\}$  to  $\mathcal{Q}_{present}$ .

*Challenge Policy Definition.*  $\mathcal{A}$  generates an acceptable hospital set  $S^*$  and sends  $S^*$  to  $\mathcal{C}$ . If  $S^*$  are included in  $\{hpk_i\}_{i \in [1, n_H]}$ ,  $\mathcal{C}$  runs Policy Definition algorithm to create  $pol^*$  and then returns  $pol^*$  to  $\mathcal{A}$ .

*Output.*  $\mathcal{A}$  outputs a  $pt^*$  and wins if

- The algorithm Present Verify  $(pt^*, pol^*)$  returns 1.
- $(pol^*) \notin \mathcal{Q}_{present}$ ;
- $\nexists (i_j, attr_{i_j}) \in \mathcal{Q}_{issue}$  such that  $pol^*$  is satisfied.

We say that a blockchain-based data trading scheme is existentially unforgeable under chosen message attack if no polynomial-time adversary can win GAME Unforgeability with a non-negligible probability.

For privacy preservation, it should be guaranteed that only the required data can be obtained by a data buyer, i.e., personal information is not leaked, which we describe as anonymity of

personal data. Moreover, since the data on smart contracts are public, the identity of a data seller should be protected, and transactions of the same data seller should not be linked, which we describe as unlinkability. The definitions of anonymity and unlinkability are listed below.

Anonymity requires that a data buyer cannot obtain the identity information of a data seller. We describe this requirement using Game Anonymity, where  $\mathcal{C}$  and  $\mathcal{A}$  are involved.

*Definition 2.2 (GAME Anonymity)*

*Setup.* Given a security parameter  $k$  and the number of hospitals  $n_H$  in the system.  $\mathcal{C}$  runs Setup algorithm to generate public parameters  $pp$ . Then, by invoking Hospital Key Generation algorithm,  $\mathcal{C}$  creates public-private key pairs  $(hpk_i, hsk_i)_{i \in [1, n_H]}$  for these hospitals.  $\mathcal{C}$  sends  $pp$  and  $\{hpk_i\}_{i \in [1, n_H]}$  to  $\mathcal{A}$ .

*Issue Queries.*  $\mathcal{A}$  can submit *Issue queries* that are the same as the issue queries in GAME Unforgeability.  $\mathcal{C}$  responds to  $\mathcal{A}$  based on the queries.

*Present Queries.*  $\mathcal{A}$  can submit *present queries* as in GAME Unforgeability.  $\mathcal{C}$  returns the results to  $\mathcal{A}$ .

*Challenge.*  $\mathcal{A}$  defines a data trading policy ( $pol$ ), which includes the public keys of acceptable hospitals and data requirements.  $\mathcal{A}$  then sends  $pol$  to  $\mathcal{C}$ .  $\mathcal{C}$  generates two public-private key pairs  $(upk_i, usk_i)_{i \in \{0,1\}}$ , and sends  $upk_0$  and  $upk_1$  to  $\mathcal{A}$ .  $\mathcal{C}$  chooses  $b \in \{0, 1\}$ , and uses  $usk_b$  creating a  $pt_b$  that satisfies  $pol$ . Then  $\mathcal{C}$  returns  $pt_b$  to  $\mathcal{A}$ .

*Guess.*  $\mathcal{A}$  outputs a guess of  $b$  for  $pt_b$ .

$\mathcal{A}$  wins the game if  $b$  is correct. The advantage of  $\mathcal{A}$  is calculated by  $Pr[b' = b] - \frac{1}{2}$ .

We say that a blockchain-based data trading scheme achieves anonymity if no polynomial-time adversary can win GAME Anonymity with non-negligible advantage.

Unlinkability requires that after a data seller uploads  $(CB, \pi_b)$  to the smart contract, no external adversary can link two data sellers together based on  $(CB, \pi_b)$ . In the security game for Unlinkability, we let  $\mathcal{A}$  outputs a public key  $\{bpk\}$  for a data buyer, a policy  $pol$ , and two sets of  $\{EMR_i, hpk_i\}_{i \in \{0,1\}}$  for different data sellers and they both satisfy  $pol$ . Then,  $\mathcal{C}$  generates  $(CB, \pi_b)$  from one of the two EMRs and sends  $(CB, \pi_b)$  to  $\mathcal{A}$ .  $\mathcal{A}$  needs to decide which EMR is used to derive  $(CB, \pi_b)$ . GAME Unlinkability is defined as follows:

*Definition 2.3 (GAME Unlinkability)*

*Setup.*  $\mathcal{A}$  runs Setup algorithm and outputs public parameters  $pp$ .

*Issue.*  $\mathcal{A}$  runs Issue algorithm and generates  $\{EMR_i, hpk_i\}_{i \in \{0,1\}}$  for different patients whose private key is  $\{usk\}_{i \in \{0,1\}}$ .  $\mathcal{A}$  also defines a public key  $\{bpk\}$  for a data buyer and a policy  $pol$ , where  $\{hpk_i\}_{i \in \{0,1\}}$  are acceptable hospitals.  $\mathcal{A}$  sends the resulting EMRs and the corresponding hospital public keys to  $\mathcal{C}$ .

*Ciphertext Generation.*  $\mathcal{C}$  runs Verify to check the validity of  $\{EMR_i, hpk_i\}_{i \in \{0,1\}}$ . It also verifies that  $\{EMR_i\}_{i \in \{0,1\}}$  satisfy  $pol$ . Then, it randomly chooses a bit  $\xi \in \{0, 1\}$  and generates  $(CB, \pi_b)$  based on  $EMR_\xi$  and  $bpk$ .  $\mathcal{C}$  then sends  $(CB, \pi_b)$  to  $\mathcal{A}$ .

*Output.*  $\mathcal{A}$  decides which EMR is used to derive  $(CB, \pi_b)$  and outputs  $\xi^*$ .  $\mathcal{A}$  wins if  $\xi^* = \xi$ .

We say that a blockchain-based data trading scheme satisfies unlinkability if no polynomial-time adversary can win GAME Unlinkability with a non-negligible probability.

#### IV. BUILDING BLOCKS

##### A. Structure-Preserving Signature

In a structure-preserving signature scheme [27], messages, the verification key, and signatures are all group elements from  $G_1$  and  $G_2$ . We recall a structure-preserving signature scheme proposed by Groth [28]. Moreover, as [29], we consider two variants of the scheme, Groth<sub>1</sub> and Groth<sub>2</sub>. Groth<sub>1</sub> is described as below, and Groth<sub>2</sub> can be attained by swapping the roles of  $G_1$  and  $G_2$ .

- Groth<sub>1</sub>.Setup( $\lambda$ ): Given a security parameter ( $\lambda$ ), the algorithm outputs public parameters ( $pp$ ), which consist of  $(G_1, G_2, G_T, p, e, g, \hat{g})$ , where  $p$  is the order of  $G_1$  and  $G_2$ .  $g$  and  $\hat{g}$  are generators of  $G_1$  and  $G_2$ , respectively.  $e : G_1 \times G_2 \rightarrow G_T$  is a computable bilinear map. The algorithm also outputs a random number ( $Y \in G_1$ );
- Groth<sub>1</sub>.KeyGen( $pp$ ): This algorithm randomly chooses an element  $u \in \mathbb{Z}_p^*$  as the private key  $sk$  and computes the corresponding public key as  $pk = U = \hat{g}^u$ ;
- Groth<sub>1</sub>.Sign( $pp, sk, M$ ): This algorithm signs a message  $M \in G_1$  using  $sk$ . It randomly selects a number  $r \in \mathbb{Z}_p^*$ , and computes  $\hat{R} = \hat{g}^r$ ,  $S = (Y \cdot g^u)^{\frac{1}{r}}$ ,  $T = (Y^u \cdot M)^{\frac{1}{r}}$ . The signature of  $M$  is  $\sigma = (\hat{R}, S, T)$ ;
- Groth<sub>1</sub>.verify( $pp, pk, \sigma, M$ ): Given message  $M$  and its signature  $\sigma$ , the algorithm checks the correctness of the signature. It outputs 1 if  $e(S, \hat{R}) = e(Y, \hat{g}) \cdot e(g, \hat{U})$  and  $e(T, \hat{R}) = e(Y, \hat{U}) \cdot e(M, \hat{g})$ ;
- Groth<sub>1</sub>.Rand( $pp, \sigma$ ): This algorithm randomizes signature  $\sigma$ . It first picks  $r' \in \mathbb{Z}_p^*$  and computes  $\hat{R}' = \hat{R}^{r'}$ ,  $S' = S^{\frac{1}{r'}}$ ,  $T' = T^{\frac{1}{r'}}$ . The randomized signature is  $\sigma' = (\hat{R}', S', T')$ .

##### B. Zero-Knowledge Proof (ZKP)

ZKP allows a prover to convince a verifier that it knows a secret without leaking the secret itself [30]. For statements related to discrete logarithms and relations in prime-order groups,  $\Sigma$ -protocols are widely utilized because of the simplicity and versatility.

$\Sigma$ -Protocols. Let  $R$  be a binary relation and  $(x, Y) \in R$ . A prover which has  $x$  and wants to prove to the verifier that it knows  $x$  such that  $(x, Y) \in R$  can interact with the verifier utilizing the following 3-move  $\Sigma$ -protocol, where  $P_1$ ,  $P_2$ , and  $V$  are three algorithms.

- 1) With the input  $(x, Y)$ , the prover randomly chooses a random number  $a$  and computes a number  $T$  by using  $P_1(x, Y, a)$ . Then, the prover sends  $T$  to the verifier while keeping  $a$  private;
- 2) The verifier randomly selects a challenge  $c$  from a challenge set  $C$ , and sends  $c$  to the prover;
- 3) The prover computes an element  $s$  using  $P_2(x, Y, a, c)$  and returns  $s$  to the verifier. The verifier runs  $V(Y, T, c, s)$ , and if the output is true, the verifier accepts the proof.

TABLE II  
AN EXAMPLE OF AN EMR

Attributes	Attribute Values
Patient Name	Alice
Gender	Female
Age	50
Address	ABCDEFGFG
Social Insurance Number	987654321
Diastolic Blood Pressure	72
Glucose	119
Tricep Skinfold Thickness	35
Serum Insulin	140
Body Mass Index	34
A Pedigree Function for Diabetes	0.46
Diagnosis	Diabetes
Medications	*****

#### V. BLOCKCHAIN-BASED FAIR AND FINE-GRAINED DATA TRADING

In this section, we first describe the overview of the proposed scheme, then present the detailed construction.

##### A. Overview

There are two phases in our proposed fair and fine-grained data trading scheme: the pre-trading phase and the data trading phase. In the pre-trading phase, a hospital generates an EMR for a patient, which includes attributes of the record, the corresponding attribute values, and the signature of the hospital. The attributes of EMR can be the *name*, *age*, *gender*, and other attributes related to the diagnosed disease. A toy example is shown in Table II. In the data trading phase, the hospital is not involved. A data buyer creates a smart contract, which defines the attribute requirements, deposits the data rewards, and transfers the rewards to the data seller if the seller proves that the requirements are satisfied and the buyer can obtain the desired data. Zero-knowledge proof techniques are utilized such that only the required attributes and attribute values are leaked to the data buyer, and other information is hidden, which achieves privacy preservation. On the other hand, the data buyer can verify whether the predefined requirements are met, and automatic execution of the smart contract can guarantee that the eligible data sellers can receive the rewards and the data buyer can obtain the data, which achieves fairness of the data trading.

In the data trading phase, a data buyer first generates a one-time encryption public-private key pair ( $bpk, bsk$ ). Then, the data buyer deploys a smart contract (SC) on the blockchain and deposits the reward in the SC. Moreover, the information on SC includes  $bpk$  and data requirements, which can specify the attribute set required and the attribute values of certain attributes. For example, a data buyer wants to gather data related to diabetes. As a result, the attribute value for the attribute *Disease* is *diabetes*, and attribute values for attributes *Disease*, *Glucose*, *Diastolic Blood Pressure*, *Serum Insulin*, *Age* are required [31]. If a data seller has the data that satisfy the requirements of the data buyer, the data seller and the data buyer act as follows: 1) The data seller proves to the data buyer that it has the required



TABLE III  
A SUMMARY OF SYMBOLS

Symbol	Definition
$(hpk, hsk)$	Public-private signing key pair of a hospital
$(upk, usk)$	Public-private encryption key pair of a patient
$(bpk, bsk)$	Public-private encryption key pair of a buyer
$\{Attr_i, D_i\}_{i \in [1, l]}$	An attribute and its attribute value
$k$	Symmetric key used to encrypt $\{D_i\}$
$(CK_1, CK_2)$	Ciphertext of $k$ encrypted with $upk$
$CD_i$	Ciphertext of $D_i$
$\sigma$	Signature of a hospital included in an EMR
$I$	Set of attributes that need to be disclosed
$J$	Set of attributes whose value need to be disclosed
$(CK'_1, CK'_2)$	Randomized $(CK_1, CK_2)$

data. To be specific, it sends the attributes in the required attribute set ( $S$ ), the signature on the attributes, the ciphertexts of attribute values, and auxiliary information to the data buyer; 2) If the data buyer is convinced that the data seller has the required data, it sends a confirmation message to the smart contract; 3) The data seller encrypts  $k$  with public key  $bpk$ , and generates a zero-knowledge proof ( $\pi_b$ ) that demonstrates the ciphertext ( $CB$ ) of  $k$  under  $upk$  and the ciphertext of  $k$  under  $bpk$  are encryptions of the same message. The data seller uploads  $CB$  and  $\pi_b$  to the smart contract; 4) The smart contract checks whether  $CB$  is valid by verifying  $\pi_b$ . If  $CB$  is valid, the smart contract records  $CB$  and sends the reward to the data seller; 5) The data buyer can decrypt  $CB$  by using its private key  $bsk$  and obtain  $k$ . Then, it uses  $k$  to decrypt the ciphertexts of the attribute values.

### B. Detailed Scheme

In the pre-trading phase, HD generates public parameters for the hospitals, and a hospital and a patient interact to generate an EMR. There are four algorithms in the pre-trading phase. Some parameters and their definitions are listed in Table III.

**Setup.** In this algorithm, HD generates the public parameters. On input a security parameter ( $k$ ) and the maximum number of attributes ( $n$ ) that can be signed in an EMR, HD generates group  $G_1, G_2$  and  $G_T$  of order  $p$  with  $\{g, Y\}$  being two generators of  $G_1$  and  $\{\hat{g}, \hat{Y}\}$  being two generators of  $G_2$ . HD also randomly chooses  $n + 4$  generators,  $Z_0, \dots, Z_{n+3}$  of  $G_1$ . Then, HD defines three hash functions:  $H : \mathbb{Z}_p^n \rightarrow G_1$ , where  $H(\vec{a}) = \prod_{i=0}^{n+3} Z_i^{A_i}$  and  $\vec{a} = \{A_0, \dots, A_{n+3}\} \in \mathbb{Z}_p^{n+4}$ ;  $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$ ; and  $\mathcal{H} : \{0, 1\}^* \rightarrow G_2$ .

The public parameters  $pp$  is  $pp = \{e, G_1, G_2, G_T, p, g, \hat{g}, Y, \hat{Y}, (Z_i)_{i=0}^{n+3}, H, \mathcal{H}, \mathcal{H}\}$ .

**Hospital Key Generation.** This algorithm is run by a hospital. The hospital generates a public-private signing key pair ( $hpk, hsk$ ) by randomly selecting  $u \in \mathbb{Z}_p$ , assigning it to  $hsk$ , and computing  $hpk = \hat{U} = \hat{g}^u$ . Then, it registers the public key with HD, which publishes the public keys of all hospitals.

**Issue.** In this algorithm, a hospital interacts with a patient to generate an EMR for the patient. Let the number of attributes in the EMR be  $l$ . For attributes  $\{Attr_i\}_{i \in [1, l]}$ , the

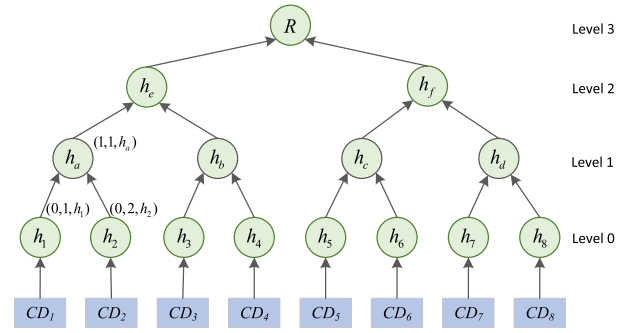


Fig. 2. Sequence-based Merkle Hash Tree.

corresponding attribute values are denoted as  $\{D_i\}_{i \in [1, l]}$ . The interaction between the patient and the hospital is as follows:

- The patient generates its public-private key pair ( $upk, usk$ ) by randomly selecting  $usk \in \mathbb{Z}_p^*$  and calculating  $upk = Z_0^{usk}$ ;
- The patient generates a zero-knowledge proof ( $\pi_u$ ) for  $usk$  and sends  $\{\pi_u, upk\}$  to the hospital;
- The hospital verifies  $\pi_u$ . If  $\pi_u$  is valid, the hospital encrypts the attribute values in the EMR. To be specific, 1) the hospital chooses a symmetric key ( $k \in \mathbb{Z}_p^*$ ), and encrypts  $k$  with  $upk$  using the ElGamal encryption algorithm. The ciphertext of  $k$  is  $CK = \{CK_1, CK_2\}$ , where  $CK_1 = g^v$ ,  $CK_2 = k \cdot upk^v$ , and  $v$  is randomly chosen from  $\mathbb{Z}_p^*$ ; 2) The hospital then encrypts  $\{D_i\}_{i \in [1, l]}$  using the AES encryption algorithm with  $k$  being the encryption key. The ciphertext of  $D_i$  is denoted as  $CD_i$ ; 3) It builds a Sequence-based Merkle Hash Tree (SMHT) based on  $\{CD_i\}_{i \in [1, n]}$ , where  $CD_i$  are the leaf nodes, and  $\{CD_i\}_{i \in [l+1, n]}$  are randomly chosen from  $\{0, 1\}^*$ . SMHT is slightly different from Merkle Hash Tree (MHT) [32], where each node is represented as three elements: (level, serial number, hash value), which denote the level of the node in the tree, the serial number of the node in that level, and the hash value of the node. In the tree, the hash value of an internal node is generated from its two child nodes. An example is shown in Fig. 2, where we denote  $\mathcal{H}(CD_i)$  as  $h_i$  and they are ordered in level 0.  $h_a = \mathcal{H}(h_1 || h_2)$ , and the node  $a$  is denoted as  $(1, 1, h_a)$ . The hash value of the root node of SMHT is  $h_R$ ; 4) The hospital also generates an SMHT based on  $\{D_i\}_{i \in [1, n]}$ . The resulting hash value of the root node is  $h_{\hat{R}}$ ;
- The hospital signs  $\{upk, \{Attr_i\}_{i \in [1, n]}, h_R, h_{\hat{R}}, CK\}$  as follows: The hospital first computes  $A_i = \mathcal{H}(Attr_i)$  for  $i \in [1, l]$ . Let  $A_i$  be 0 for  $i \in [l+1, n]$ ,  $A_{n+1} = h_R$ ,  $A_{n+2} = h_{\hat{R}}$ , and  $A_{n+3} = \mathcal{H}(CK_1 || CK_2)$ . Then, the hospital randomly chooses  $r \in \mathbb{Z}_p^*$ , and computes  $H(\vec{a}) = upk \cdot \prod_{i=1}^{n+3} Z_i^{A_i}$ ,  $\hat{R} = \hat{g}^r$ ,  $S = (Y \cdot g^u)^{1/r}$ , and  $T = (Y^u \cdot H(\vec{a}))^{1/r}$ . The resulting signature is  $\sigma = (\hat{R}, S, T)$ . The hospital sends EMR, which includes  $\{CK, \{Attr_i, CD_i\}_{i \in [1, n]}, h_R, h_{\hat{R}}, \sigma\}$ , to the patient.

**Verify.** In this algorithm, the patient verifies the validity of the received EMR. The patient first verifies the correctness of  $h_R$  by re-generating a SMHT based on  $\{CD_i\}_{i \in [1, n]}$  and checks whether the hash value of the root is  $h_R$ . If  $h_R$  is

valid, the patient checks the correctness of  $\sigma$ . To be specific, the patient calculates  $A_i = \mathcal{H}(Attr_i)$  for  $i \in [1, l]$ . Let  $A_i = 0$  for  $i \in [l+1, n]$ ,  $A_{n+1} = h_{\mathcal{R}}$ ,  $A_{n+2} = h_{\hat{\mathcal{R}}}$ , and  $A_{n+3} = \mathcal{H}(CK_1 || CK_2)$ . Then, the patient checks whether  $e(S, \hat{R}) = e(Y, \hat{g}) \cdot e(G, \hat{U})$  and  $e(T, \hat{R}) = e(Y, \hat{U}) \cdot e(H(\vec{a}), \hat{g})$  hold. If  $\sigma$  is valid, the patient can decrypt  $CK$  using  $usk$  and obtain  $k$ . Then, it decrypts  $CD_i$  by utilizing  $k$ . Based on  $D_i$ , the patient builds an SMHT and checks whether  $A_{n+2} = h_{\hat{\mathcal{R}}}$ . If it is correct, the patient keeps the EMR locally, which is  $\{CK, \{Attr_i, CD_i\}_{i \in [1, n]}, h_{(\mathcal{R})}, h_{(\hat{\mathcal{R}})}, \sigma\}$ .

In the data trading phase, a data seller, which is a patient in the pre-trading phase, a data buyer, and a smart contract interact to complete data trading. A data buyer first creates a smart contract, which defines the data requirements, its public key, and the reward of data. For data requirements, the data buyer can specify which attributes and attribute values need to be disclosed and the hospitals it accepts. If a data seller has the data that satisfy the requirements, it can trade data with the data buyer in the data trading phase. This phase includes five algorithms, which are listed as follows:

**Policy Definition.** In this algorithm, the data buyer generates its public key and private key. Then, it defines the hospitals it accepts and the attribute requirements for the desired data. The data buyer first randomly selects  $w \in \mathbb{Z}_p^*$ , and calculates  $W = g^w$ . The public-private key pair of the data buyer is  $(bpk, bsk) = (W, w)$ .

Then, let the set of the public keys of the acceptable hospitals be  $\mathcal{S} = \{hpk_1, \dots, hpk_\eta\}$ . For  $i \in [1, \eta]$ , the buyer signs the public keys  $hpk_i$  by choosing a random  $\xi_i$  and computing  $R_i = g^{\xi_i}$ ,  $\hat{S}_i = (\hat{Y} \cdot \hat{g}^w)^{\frac{1}{\xi_i}}$ ,  $\hat{T}_i = (\hat{Y}^w \cdot hpk_i)^{\frac{1}{\xi_i}}$ . The signatures  $\{\sigma_i\}_{i \in [1, \eta]}$  for  $\{hpk_i\}$  are  $\{R_i, \hat{S}_i, \hat{T}_i\}$ .

The data buyer then defines the set  $I$  to be the attributes that need to be disclosed and  $J$  to be the attributes whose attribute values need to be leaked to the data buyer before the payment. Let  $JV$  be a set that specifies the attribute values corresponding to attributes in  $J$ . For example, if the data buyer wants to collect data related to diabetes and data should include values for the attributes {Disease, Age, Diastolic Blood Pressure, Serum Insulin}, it specifies  $I = \{\text{Disease, Age, Diastolic Blood Pressure, Serum Insulin}\}$ ,  $J = \{\text{Disease}\}$ , and  $JV = \{\text{Diabetes}\}$ . By disclosure of the attributes in  $I$  and attribute values ( $JV$ ) of attributes in  $J$ , the data buyer learns that the data seller has the data it needs. The data buyer then computes the hashes of  $I$  and  $\{J, JV\}$  by using hashing function  $H$  and signs them.

Considering that the size of  $\mathcal{S}$  may be large, the data buyer can store  $\mathcal{S}$  and  $\{\sigma_i\}_{i \in [1, \eta]}$  at an off-chain storage platform and uploads the storage position,  $p, g, Y, \hat{g}, \hat{Y}, bpk, I, J, JV$ , and their signatures to the smart contract. The defined policy  $pol$  is  $pol = (S, I, J, JV)$ . The data buyer also deposits the data reward to the smart contract.

**Policy Verification.** In this algorithm, a data seller verifies the validity of the policy uploaded by the data buyer and the correctness of the signatures for public keys in  $\mathcal{S}$ . Let  $hpk_j = hpk$ , which is the hospital that issues an EMR to the data seller. Given  $hpk_j$  and  $\{R_j, \hat{S}_j, \hat{T}_j\}$ , the data seller

verifies whether  $e(R_j, \hat{S}_j) = e(g, \hat{Y}) \cdot e(W, \hat{g})$  and  $e(R_j, \hat{T}_j) = e(W, \hat{Y}) \cdot e(g, hpk_j)$ . If the equations hold, the signatures for public keys in  $\mathcal{S}$  are valid. The data seller also verifies the signatures of  $I$  and  $\{J, JV\}$ . If they are valid, the policy defined by the data buyer is verified.

**Present.** In this algorithm, the data seller generates a zero-knowledge proof that proves it has an EMR that satisfies the defined policy. Denote  $\Pi$  as the set of indexes of the attributes in  $I$  and  $\Omega$  as the set of indexes of the attributes in  $J$ .

Given the policy  $pol = (S, I, J, JV)$ , the data seller generates a zero-knowledge proof as follows:

- To hide the identity of the hospital, it randomizes the signature  $(\sigma_j = (R_j, \hat{S}_j, \hat{T}_j))$  on the hospital. That is, it randomly selects  $r'' \in \mathbb{Z}_p^*$ , and computes  $R'_j = R_j^{r''}$ ,  $\hat{S}'_j = \hat{S}_j^{\frac{1}{r''}}$ ,  $\hat{T}'_j = \hat{T}_j^{\frac{1}{r''}}$ ;
- It generates a new ciphertext for  $k$  by choosing a random  $\delta \in \mathbb{Z}_p^*$  and computing  $CK'_1 = CK_1^\delta$  and  $CK'_2 = CK_2 \cdot upk^\delta$ . Denote  $CK' = \{CK'_1, CK'_2\}$ . The data seller also generates a commitment of  $usk$  by choosing  $\epsilon$  and calculating  $B = g^{usk} Y^\epsilon$ ;
- It chooses random values  $\alpha, \beta, \gamma, \phi \in \mathbb{Z}_p^*$  and computes the blinded signature  $(\sigma_{attri})$  on  $H(\vec{a})$  under  $hpk$  as  $(\hat{R}', S', T') = (\hat{R}, S^{\frac{1}{\alpha}}, T^{\frac{1}{\beta}})$ . Then, it computes the blinded public key of the hospital  $(hpk'_j)$  as  $hpk_j^{\frac{1}{\gamma}}$ . After that, it calculates the blinded signature  $(\sigma_{hpk})$  on  $hpk_j$  under the data buyer's public key  $bpk$  as  $(R'_j, \hat{S}'_j, \hat{T}'_j) = (R'_j, \hat{S}'_j, \hat{T}'_j)^{\frac{1}{\phi}}$ ;
- The data seller generates a zero-knowledge proof  $\pi_d$  with the witness  $(\alpha, \beta, \gamma, \phi, usk, \{A_i\}_{i \notin \Pi}, \epsilon)$ . The statement for  $\pi_d$  is listed as follows:

$$e(S', \hat{R}')^\alpha = e(Y, \hat{g}) \cdot e(g, hpk'_j)^\gamma \quad (1)$$

$$\wedge e(T', \hat{R}')^\beta = e(Y, hpk'_j)^\gamma \cdot e\left(Z_0^{usk} \prod_{i=1}^{n+3} Z_i^{A_i}, \hat{g}\right) \quad (2)$$

$$\wedge e(R'_j, \hat{S}'_j) = e(g, \hat{Y}) \cdot e(bpk, \hat{g}) \quad (3)$$

$$\wedge e(R'_j, \hat{T}'_j)^\phi = e(bpk, \hat{Y}) \cdot e(G, hpk'_j)^\gamma \quad (4)$$

$$\wedge CK'_2 / CK_2 = (CK'_1 / CK_1)^{usk} \quad (5)$$

$$\wedge B = g^{usk} Y^\epsilon \quad (6)$$

In the statement, (1) and (2) check the validity of the signature on  $H(\vec{a})$ , and (3) and (4) verify the validity of the signature on  $hpk$ . Equation (5) guarantees the consistency of the ciphertexts of  $k$ . Equation (6) demonstrates that  $B$  is a Peterson commitment [33] for  $usk$ .

The data seller then generates the auxiliary authentication information (AAI) for  $\{CD_i\}_{i \in \Pi}$ . The AAI for leaf node  $i$  contains the sibling nodes from  $i$  to the root node. For example, the AAI for leaf node 1 includes the nodes  $\{2, b, f\}$ , i.e.,  $\{(0, 2, h_2), (1, 2, h_b), (2, 2, h_f)\}$ . To construct the AAI for leaf nodes  $\{i\}_{i \in \Pi}$ , the data seller first generates AAI for the nodes whose indexes are in  $\Pi$ , and the resulting set is called  $AAI_\Pi$ . Then, the data seller deletes the ancestor nodes of leaf nodes



$\{i\}_{i \in \Pi}$  from  $AAI_{\Pi}$ . Note that from  $\{CD_i\}_{i \in \Pi}$  and  $AAI_{\Pi}$ , we can recover  $h_{\mathcal{R}}$ . Similarly, the data seller generates  $AAI_{\Omega}$  for  $\{D_i\}_{i \in \Omega}$ .

The data seller also generates a one-time public-private address for blockchain transaction and registers its public address ( $PA$ ) on the smart contract. Denote  $pt$  as  $pt = (\sigma_{attri}, hpk'_j, \sigma_{hpk}, \pi_d)$ . The data seller sends  $(pt, \{CD_i, (i, Attri_i)\}_{i \in \Pi}, AAI_{\Pi}, AAI_{\Omega}, h_{\mathcal{R}}, h_{\hat{\mathcal{R}}}, CK, CK', \{D_j\}_{j \in J}, PA)$  to the data buyer.

*Present Verify.* In this algorithm, the data buyer verifies whether the data of the data seller satisfy the requirements. Given  $pt$ , and  $(i, Attri_i)_{i \in \Pi}$ , the data buyer sets  $A_{n+1} = h_{\mathcal{R}}$ ,  $A_{n+2} = h_{\hat{\mathcal{R}}}$ , and  $A_{n+3} = H(CK_1 || CK_2)$ . Then, it verifies  $\pi_d$ . If  $\pi_d$  is valid, the data buyer ensures that  $\sigma_{attri}$  is valid, and it is derived from a signature generated from an acceptable hospital.

Then, the data seller verifies whether  $\{CD_i\}$  matches with the  $\{Attri_i\}$ , i.e., they have the same order in EMR. The data seller recomputes the hash value of the root ( $h_{\mathcal{R}'}$ ) of SMHT with  $\{CD_i\}$  and  $AAI_{\Pi}$ . If  $h_{\mathcal{R}} = h_{\mathcal{R}'}$ , The positions of  $\{CD_i\}$  are ensured. The serial numbers of  $\{Attri_i\}$  are verified by checking the validity of  $\pi_d$ . The data seller then verifies the validity of  $D_i$  by recomputing the  $h_{\hat{\mathcal{R}'}}$  based on  $D_i$  and  $AAI_{\Omega}$ . If  $h_{\hat{\mathcal{R}}} = h_{\hat{\mathcal{R}'}}$ ,  $\{D_i\}_{i \in \Omega}$  are valid.

Then, given the ciphertext  $CK'$ , the commitment  $B$ , the data buyer uploads a confirmation message  $CM = \{CK', B\}$  to the smart contract.

*Ciphertext Generation.* In this algorithm, the data seller encrypts  $k$  under the data buyer's public key  $bpk$  and generates a zero-knowledge proof that proves the ciphertext is created correctly. To be specific, the data seller randomly chooses  $t \in \mathbb{Z}_p^*$ , and computes  $CB_1 = g^t$  and  $CB_2 = k \cdot bpk^t$ . Denote  $CB = \{CB_1, CB_2\}$ . Notice that if  $(CB_1, CB_2)$  and  $(CK'_1, CK'_2)$  are two encryptions of  $k$ ,  $CB_2/CK'_2 = CK'_1^{-usk} bpk^t$ . Thus, the data seller generates a zero-knowledge proof  $\pi_b$ , which proves that  $PK : \{(usk, t, \epsilon) : CB_2/CK'_2 = CK'_1^{-usk} bpk^t \wedge CB_1 = g^t \wedge B = g^{usk} Y^{\epsilon}\}$ . The details of generating  $\pi_b$  are as follows:

- The data seller randomly chooses  $\rho_1, \rho_2, \rho_3 \in \mathbb{Z}_p^*$ , and computes  $T_1 = CK'_1^{-\rho_1} \cdot bpk^{\rho_2}$ ,  $T_2 = g^{\rho_2}$ , and  $T_3 = g^{\rho_1} Y^{\rho_3}$ ;
- The data seller computes  $c_1 = \mathcal{H}(T_1 || T_2 || T_3)$  as a challenge;
- The data seller computes  $z_1 = \rho_1 - c_1 \cdot usk$ ,  $z_2 = \rho_2 - c_1 \cdot t$ , and  $z_3 = \rho_3 - c_1 \cdot \epsilon$ .

The resulting  $\pi_b$  includes  $\{T_1, T_2, T_3, c_b, z_1, z_2, z_3\}$ . The data seller then uploads  $\pi_b$  and  $CB$  to the smart contract. Given  $\{CB, CK', B, g, Y, bpk\}$ , the validators of the blockchain verify whether the equation

$$c_b = H(CK'_1^{-z_1} bpk^{z_2} \cdot (CB_2/CK'_2)^{c_1} || g^{z_2} CB_1^{c_1} || g^{z_1} Y^{z_3} B^{c_1})$$

holds. If  $\pi_b$  is valid, the smart contract transfers the data reward to the data seller.

Note that the storage scalability is out of the scope of this article. For the storage scalability issue, one may adopt multiple-node cooperative storage [34], data reduction [35], or sharding [36] methods to reduce the storage pressure on storage

nodes. For the block structure, we follow the conventional block structure, which contains a block header and a block body. The former includes the whole information of a block, such as hash values of the previous block and the current block, tree root, timestamp, nonce, and information of the consensus protocol. The block body includes all transaction records, which are organized in a tree structure [37].

## VI. SECURITY PROOF

In this section, we prove that the proposed scheme achieves fairness and privacy preservation.

### A. Fairness

As mentioned in Section III, a blockchain-based data trading scheme achieves fairness if atomicity, data ownership, and data availability are satisfied. The atomicity of transactions can be guaranteed by the smart contract. We use unforgeability to formally describe the data ownership and data availability. In the following, we prove that our scheme achieves unforgeability.

*Theorem 1.* If  $\text{Groth}_1$  and  $\text{Groth}_2$  are existential unforgeable, hash function  $H$  is collision-resistant, and the NIZK is simulation-sound extractable and zero-knowledge, our proposed scheme satisfies unforgeability.

In our scheme, an adversary can break the unforgeability by either forging a signature on a hospital  $hpk_j$  that is not included in the accepted hospital set defined in  $pol$ ; Or the adversary can forge a signature on an EMR that is supposed to be signed by an honest hospital in  $pol$ , and the signature can be verified by Verify algorithm. In the following, we will prove that Theorem 1 holds.

*Proof.* We prove the unforgeability property by a series of games, which are listed as follows.

- *Game<sub>0</sub>* : *Game<sub>0</sub>* is the same as GAME Unforgeability, where a challenger  $\mathcal{C}$  and an adversary  $\mathcal{A}$  are involved;
- *Game<sub>1</sub>* : In *Game<sub>1</sub>*, we modify the way to answer present queries, where  $\pi_d$  is generated by invoking a zero-knowledge and extractable NIZK simulator. Because the NIZK is zero-knowledge, *Game<sub>1</sub>* and *Game<sub>0</sub>* are indistinguishable.

In *Game<sub>1</sub>*,  $\mathcal{A}$  wins if  $\text{Present Verify}(pt^*, pol^*) = 1$ . Given  $pt^*$ , we can utilize the NIZK extractor to extract the witness  $(\alpha, \beta, \gamma, \phi, usk, \{A_i\}_{i \in \Pi}, \epsilon)$ . Note that  $usk$  is the private key of the data seller, which denotes the data ownership. Then, with the witness, we can obtain signature  $\sigma$ , signature  $\sigma_j$ ,  $usk$ ,  $hpk$ , and  $\{Attri_i\}_{i \in [1, n]}$ . Let  $F$  be the event that the extractor fails to extract a witness. Let  $NF_1$  be the event that  $\mathcal{A}$  wins the game and the extractor extracts a valid witness, but the obtained  $hpk$  is not included in  $pol^*$ . Let  $NF_2$  be the event that  $\mathcal{A}$  wins the game, the extractor extracts a witness, and the recovered  $hpk$  is indeed in  $pol^*$ .

Note that  $Pr[\mathcal{A} \text{ wins}] = Pr[\mathcal{A} \text{ wins} \wedge F] + Pr[NF_1] + Pr[NF_2] \leq Pr[F] + Pr[NF_1] + Pr[NF_2]$ . Since that the  $\Sigma$  NIZK protocol is extractable,  $Pr[F]$  is negligible. Next, we show that  $Pr[NF_1]$  and  $Pr[NF_2]$  are also negligible.

*Case1:* In this case,  $\mathcal{A}$  wins and the NIZK extractor extracts a witness, but  $hpk$  is not included in  $pol^*$ . In the following

$Game_a$ , we demonstrate that if case 1 happens, we can construct another adversary  $\Lambda$  that can break the existential unforgeability of Groth<sub>2</sub>.

- Given a security parameter  $k$ ,  $\Lambda$  generates  $pp$ , which is sent to  $\mathcal{A}$ .  $\Lambda$  can access a signing oracle  $\mathcal{O}_{sign}$  of Groth<sub>2</sub>. When receiving a message,  $\mathcal{O}_{sign}$  can return a signature of the message;
- $\Lambda$  acts as  $\mathcal{C}$  in the GAME Unforgeability, except that  $\Lambda$  obtains the signatures of the acceptable hospitals by asking  $\mathcal{O}_{sign}$ . That is,  $\Lambda$  runs Hospital Key Generation algorithm to generate the public-private key pairs  $(hpk_i, hsk_i)$  for  $i \in [1, n_H]$ . Then, it answers the issue queries and present queries honestly by using  $(hsk_i)$  for  $i \in [1, n_H]$ ;
- After receiving  $S^*$  from  $\mathcal{A}$ ,  $\Lambda$  submits the signing queries of  $\{(hsk_i)\}_{i \in S^*}$  for  $i \in [1, n_H]$  to  $\mathcal{O}_{sign}$ . When obtaining the signatures  $\{\sigma_i\}_{i \in S^*}$ ,  $\Lambda$  includes them in  $pol^*$ . Then,  $\Lambda$  returns  $pol^*$  to  $\mathcal{A}$ , and continues to answer issue and present queries from  $\mathcal{A}$ ;
- $\Lambda$  aborts the game if  $\mathcal{A}$  outputs a valid  $pt^*$  and wins. Since one of conditions that  $\mathcal{A}$  wins is that  $pol^* \notin \mathcal{Q}_{present}$ ;
- From  $pt^*$ ,  $\Lambda$  extracts a witness  $(\alpha, \beta, \gamma, \phi, usk, \{A_i\}_{i \notin \Pi}, \epsilon)$ . If the obtained  $hpk_j$  is not included in  $pol^*$ ,  $\Lambda$  outputs  $hpk_j$  and its signature  $\sigma_j$  as a valid forgery of Groth<sub>2</sub> signature, and wins  $Game_a$ .

In the above game,  $Pr[\Lambda \text{ wins } Game_a] = Pr[NF_1]$ . Since Groth<sub>2</sub> is existentially unforgeable under the discrete logarithm assumption [28] and the hash function  $H$  is collision-resistant,  $Pr[NF_1]$  is negligible.

**Case2:** In this case,  $\mathcal{A}$  wins the game, the extractor extracts a witness and the extracted  $hpk$  is included in  $pol^*$ .

In the following  $Game_b$ , we demonstrate that if case 2 happens, we can construct an adversary  $\Gamma$  that can break the existential unforgeability of Groth<sub>1</sub>.

- $\Gamma$  takes as input a security parameter  $k$  and generates public parameters  $pp$ . Then,  $\Gamma$  sends  $pp$  to  $\mathcal{A}$ .  $\Gamma$  can access a Groth<sub>1</sub> signing oracle  $\mathcal{O}'_{sign}$ , which will return a Groth<sub>1</sub> signature when receiving a message to be signed;
- $\Gamma$  randomly chooses  $\tau$  from  $[1, n_H]$ , and sets  $hpk_\tau$  as  $hpk$ . Then,  $\Gamma$  runs Hospital Key Generation algorithm to generate the public-private key pairs  $(hpk_i, hsk_i)$  for  $i \in [1, n_H] \setminus \tau$ ;
- $\Gamma$  answers the issue queries and present queries from  $\mathcal{A}$  as follows: For issue queries, if  $i_j = k$ ,  $\Gamma$  generates a signature of  $hpk_\tau$  by using  $\mathcal{O}'_{sign}$ . For issue queries in which  $i_j \neq k$ ,  $\Gamma$  runs Issue algorithm with  $hsk_{i_j}$  being an input and obtains an EMR. Then,  $\Gamma$  sends the resulting EMR to  $\mathcal{A}$ . For Present queries,  $\Gamma$  creates a valid  $pt$  by using the NIZK simulator;
- $\mathcal{A}$  generates an acceptable hospital set  $S^*$  and submits it to  $\Gamma$ . After receiving the  $S^*$ ,  $\Gamma$  creates  $pol^*$  by running Policy Definition algorithm and returns  $pol^*$  to  $\mathcal{A}$ .  $\Gamma$  continues to answer issue and present queries as above;
- If  $\mathcal{A}$  outputs a  $pt^*$  and the winning conditions in GAME Unforgeability are satisfied,  $\Gamma$  aborts.  $\Gamma$  extracts a witness from  $pt^*$ . If the obtained  $hpk = hpk_\tau$ ,  $\Gamma$  wins  $Game_b$  and outputs a Groth<sub>1</sub> forgery, which is  $attri_\tau$  and the extracted signature on it.

Note that if  $NF_2$  happens and  $hpk = hpk_\tau$ , according to the soundness of the NIZK extractor, the signature of  $attri_\tau$  is a valid Groth<sub>1</sub> signature. The winning conditions of  $\mathcal{A}$  guarantee that the signature is created by  $\mathcal{A}$  and is a valid forgery. From  $Game_b$ ,  $Pr[\Gamma \text{ wins}] \leq Pr[NF_2 \wedge hpk = hpk_\tau] = \frac{1}{n_H} \cdot Pr[NF_2]$ . Since Groth<sub>2</sub> is existentially unforgeable under the discrete logarithm assumption,  $Pr[NF_2]$  is negligible.

## B. Privacy Preservation

As discussed in Section III, for privacy preservation, anonymity and unlinkability should be satisfied. In the following, we prove that our scheme achieves anonymity and unlinkability, respectively.

**Theorem 2.** If ElGamal encryption is indistinguishable under the chosen plaintext attacks (IND-CPA), the Peterson commitment scheme is hiding, and the NIZK is zero-knowledge, our proposed scheme satisfies anonymity.

*Proof.* We prove the anonymity property by a series of games, which are listed below.

- **Game 0:** This game is the same as GAME Anonymity defined in Section III;
- **Game 1:** As Game 0, but we modify the Setup algorithm by using a simulator  $S_1$  to generate the common reference string  $crs$  and a trapdoor  $\tau$  for the zero-knowledge proof scheme. Furthermore, we use a simulator  $S_2$  of the zero-knowledge proof scheme to generate  $\pi_d$ . We can observe that Game 1 and Game 0 are indistinguishable if the zero-knowledge proof scheme for the DDH tuple has zero-knowledge property;
- **Game 2:** As Game 1, but we modify  $CK_1, CK_2, CK'_1, CK'_2$ , which are results of ElGamal encryption, by choosing four random numbers in Group  $G$ . Since the ElGamal encryption is IND-CPA secure, Game 2 and Game 1 are indistinguishable;
- **Game 3:** As Game 2, but we modify the commitment  $B$  by randomly choosing a number in Group  $G$ . Since the Peterson commitment satisfies hiding property, the probability of distinguishing Game 3 and Game 2 is a negligible number.

In the Game 3,  $\mathcal{A}$  can guess  $b$  correctly with probability  $\frac{1}{2}$ , i.e., the advantage of  $\mathcal{A}$  in Game 3 is 0. To sum up,  $\mathcal{A}$  has a negligible advantage in GAME Anonymity.

**Theorem 3.** If ElGamal encryption is IND-CPA and the NIZK is zero-knowledge, our proposed scheme satisfies unlinkability.

*Proof.* We prove the unlinkability property by a series of games, which are listed as follows. In the games, we define the successful event in game  $G_i$  as  $SE_i$ .

- **Game 0:** This game is the same as GAME Unlinkability defined in Section III;
- **Game 1:** Game 1 is almost identical to Game 0, except that in the Ciphertext Generation algorithm,  $\mathcal{C}$  replaces  $CB$ , which is the result of ElGamal encryption, with two random values in Group  $G$ . Since the ElGamal encryption is IND-CPA secure, the probability of distinguishing Game 1 and Game 0 is restricted by a negligible probability  $\epsilon_1$ , i.e.,  $|Pr[SE_1] - Pr[SE_0]| \leq \epsilon_1$ ;

- *Game 2*: Game 2 is almost identical to Game 1. The difference between the two games is that for the Setup algorithm, we use a simulator  $S_1$  to generate the common reference string  $crs$  for the zero-knowledge proof scheme as well as a trapdoor ( $\tau$ ). Moreover, in the Ciphertext Generation algorithm, we use a simulator  $S_2$  to generate proof  $\pi_b$  using  $crs$  and  $\tau$ . If the zero-knowledge proof for the DDH tuple is zero-knowledge, Game 2 and Game 1 are indistinguishable with a negligible probability  $\epsilon_2$ .

Since in Game 2, the view of  $\mathcal{A}$  is independent of  $b$ , the advantage for  $\mathcal{A}$  in Game 2 is 0. Taking all games together, the advantage for the  $\mathcal{A}$  in Game 0 is that  $Pr[SE_0] \leq \epsilon_1 + \epsilon_2$ , which is a negligible number. Thus, our scheme achieves unlinkability.

## VII. PERFORMANCE EVALUATION

In this section, we first analyze the computation, communication, and storage complexity of the proposed scheme. We also compare the computational complexity of our scheme with [12]. Then, we conduct simulations to measure the performance of the proposed scheme.

### A. Complexity Analysis

In our scheme, five entities are involved, which include the health department (HD), a hospital, a patient (the data seller), a data seller, and a smart contract. In the pre-trading phase, HD generates the public parameters that can be used by other entities, and the patient and the hospital interact to create an EMR for the patient. HD and the hospital do not participate in the data trading phase. In the following, we analyze the overheads of different entities in terms of computation, communication, and storage. For simplicity, we denote the exponentiation operation in  $G_1$ ,  $G_2$ , and  $G_T$  as  $E_1$ ,  $E_2$ ,  $E_T$ . Note that when the operation is on the elliptic curve,  $E_1$  and  $E_2$  mean scalar multiplication operation.  $M_T$  denotes a multiplication in  $G_T$ , and  $P$  represents a pairing operation. We omit the computational cost of hash functions and symmetric encryption. The sizes of an element in  $Z_p$ ,  $G_1$ ,  $G_2$ , and  $G_T$  are represented as  $|Z_p|$ ,  $|G_1|$ ,  $|G_2|$ ,  $|G_T|$ .

*Computation Overhead.* In the pre-trading phase, HD generates the public parameters in the setup algorithm. Let  $n$  be the maximum number of attributes that can be signed in an EMR. The hospital creates its public-private key pair and is involved in the issuing process. The computational overhead for the hospital is at most  $(l + 11)E_1 + 2E_2$ , where  $l$  is the number of attributes that are included in the EMR. For the patient, it needs to generate its public-private key pair and a zero-knowledge proof ( $\pi_u$ ) for its private key. Also, after receiving the EMR, the patient verifies the signature in EMR and decrypts the ciphertext  $CK$ . The computational overhead for the patient is  $3E_1 + 2M_T + 6P$ .

In the trading phase, the data buyer needs to define a policy for the data trading and verify whether a data seller has data that satisfy the policy. Let the number of acceptable hospitals for the data buyer is  $\eta$ , the computational overhead for the data buyer is  $(n + \eta + 12)E_1 + (4\eta + 8)E_2 + 9E_T + 8M_T + 13P$ . For the data seller, it first verifies the policy published by the data buyer,

TABLE IV  
COMPUTATIONAL OVERHEAD IN DATA TRADING PHASE

Computational Cost	This work	[12]
Data seller	$(n - v + 23)E_1 + 5E_2 + 6E_T + 6M_T + 13P$	$16E_1 + 1E_2$
Data buyer	$(n + \eta + 12)E_1 + (4\eta + 8)E_2 + 9E_T + 8M_T + 13P$	$\frac{9E_1 + 1E_T}{v} + 2M_T + 6P$
Smart contract	$8E_1$	$10E_1$

which includes checking the signatures on the hospital and the roots of MHTs. Moreover, the data seller needs to prove to the data seller the availability of its data and generates a ciphertext for symmetric key  $k$  and zero-knowledge proof  $\pi_b$ . Let  $v$  be the number of elements in set  $I$ . In this phase, the computational overhead for the data seller is  $(n - v + 23)E_1 + 5E_2 + 6E_T + 6M_T + 13P$ . For the smart contract, the computation involved on-chain is the verification of proof  $\pi_b$ , which causes  $8E_1$  computational overhead. Table IV shows the comparison of computational overhead in the data trading phase for this work and [12] in terms of the data seller, data buyer, and the smart contract. In the table,  $SV$  denotes the computational cost of the signature verification for each attribute and its attribute value. From the table, we can see that our scheme has slightly larger off-chain computational overheads for the data seller and data buyer than [12]. However, in our scheme, a hospital only needs to sign an EMR once, instead of generating signatures for each attribute based on the requirements of a data buyer. Moreover, our scheme can achieve fine-grained data selling with lower storage overhead for data sellers, since they do not need to store all the signatures. In addition, the proposed scheme has a lower on-chain computational cost.

*Communication Overhead.* In the pre-trading phase, the patient needs to send its public key and  $\pi_u$  to the hospital, which results in  $2|Z_p| + 2|G_1|$  communication overhead. For the hospital, besides the original EMR, which contains  $\{Attr_i, CD_i\}_{i \in [1, n]}$ , it also sends the ciphertext of  $k$ , two roots of MHTs, and signature  $\sigma$  to the patient. Thus, the communication overhead of the hospital is  $4|G_1| + 1|G_2| + 2|Z_p|$  as well as the size of the original EMR ( $EMR_O$ ).

In the data trading phase, after the data buyer defines the trading policy, it transmits the public keys of acceptable hospitals and the signatures on the public keys, set  $I$ , and  $\{J, JV\}$  to an off-chain storage platform. The public key of the data buyer and signatures on set  $I$  and set  $J$  are uploaded on the smart contract. In addition, after the data buyer validates the data availability of a data seller, it needs to send a confirmation message to the smart contract. Let  $|I|$  and  $|J|$  be the size of data in set  $I$  and  $J$ . The communication overhead for the data buyer in this phase is  $(\eta + 8)|G_1| + (3\eta + 4)|G_2| + 1|Z_p|$ , where  $\eta$  is the number of acceptable hospitals. For the data seller, after the policy verification, the data seller sends  $pt$  and the auxiliary data to the data buyer off-chain. When a confirmation message is uploaded to the smart contract, the data seller uploads ciphertext  $CB$  and zero-knowledge proof  $\pi_b$  to the smart contract. In sum, besides  $\{CD_i, (i, Attr_i)\}_{i \in \Pi}$  and  $\{D_j\}_{j \in J}$ , the communication overhead for the data seller is  $17|G_1| + 4|G_2| + 2|G_T| + (2\log n + 13)|Z_p|$ .



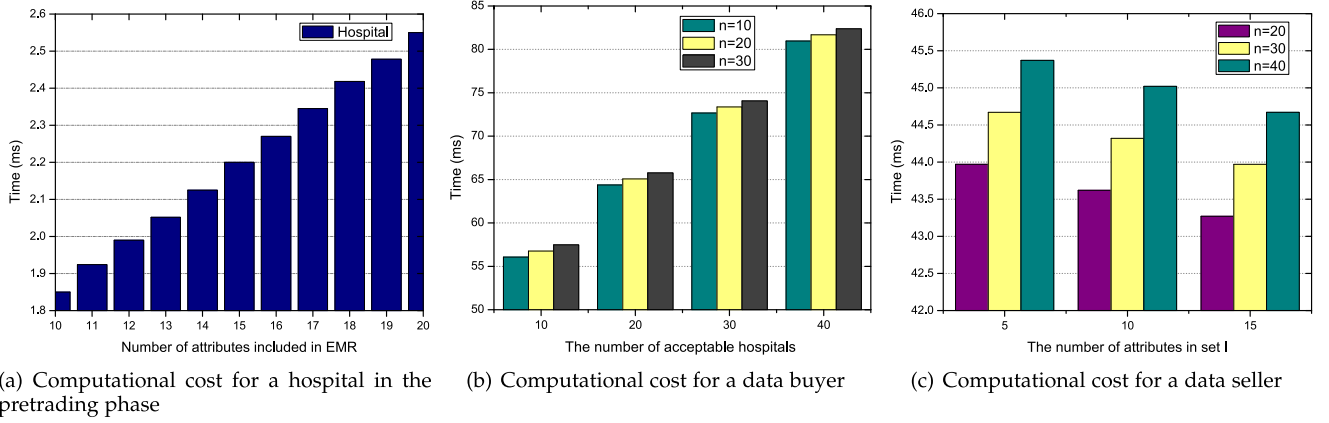


Fig. 3. Computational costs for a hospital, a data buyer, and a data seller.

TABLE V  
SIMULATION PARAMETERS

Parameter	Value
Security parameter	128
Curve	Barreto-Naehrig
Number of attributes in an EMR ( $n$ )	10-40
Number of accepted hospitals in a policy ( $\eta$ )	10-40
Number of attributes in set I ( $v$ )	5-15

**Storage Overhead.** In the pre-trading phase, HD needs to store the public parameters that include  $n + 6$  elements in  $G_1$  and 2 elements in  $G_2$ , which results in a storage cost of  $(n + 6)|G_1|$  and  $2|G_2|$ . For the hospital, the storage overhead includes the public-private key pair, the original EMR, signature  $\sigma$ , and the key  $k$ . The storage overhead for the hospital is  $3|Z_p| + 2|G_1| + 3|G_2| + |EMR_O|$ . The patient needs to store its public-private key pair and the EMR, which causes  $2|Z_p| + 6|G_1| + 1|G_2| + |EMR_O|$ .

In the data trading phase, the data seller stores the public key of the data buyer and the signature on the hospital. The storage overhead for the data seller is  $2|G_1| + 2|G_2|$ . For the data buyer, it saves its public-private key pair, the ciphertext of key  $k$  and required data, which is about  $4|G_1| + |CD|$ , where  $|CD|$  is the size of the ciphertexts  $\{CD_i\}_{i \in \Pi}$ . For the smart contract, it maintains zero-knowledge proof  $\pi_b$  and elements that are useful for verifying the proof. The storage overhead for the smart contract is  $5|Z_p| + 13|G_1| + 4|G_2|$ .

### B. Simulation Results

We conduct simulations for our proposed scheme and evaluate its performance on a notebook with Intel(R) Core(TM) i5-1135G7 @ 2.40 GHz. The RAM is 16 GB. We employ the Miracl library [38] to measure the time costs of pairing-based cryptographic operations. We instantiate the asymmetric prime-order groups with the Barreto-Naehrig (BN) elliptic curve [39]. The security parameter is set as 128. The exponentiations for generators in  $G_1$  and  $G_2$  are precomputed in our simulations. We omit the computational cost for hash functions. The simulation parameters are listed in Table V.

In the simulations, we test the computational cost for the hospital, the patient, and the data buyer by measuring their cryptographic operations. In the pretrading phase, the patient needs to generate its public-private key pair, generate a zero-knowledge proof for its private key, and verify the validity received from the hospital. The computational overhead for the patient is about 15.63 ms. Fig. 3(a) shows the overhead of the issue algorithm on the hospital side. We can see that the computational overhead for the hospital increases when the number of attributes included in the EMR grows. In particular, when an EMR contains 20 attributes, the issuing cost for the hospital is 2.55 ms.

In the trading phase, the data buyer first defines data trading policies in Policy Definition algorithm, where signatures for the acceptable hospitals' public keys are required. The data buyer also needs to verify the data availability of a data seller. The computational overhead of the data buyer is shown in Fig. 3(b), where the number of acceptable hospitals ( $\eta$ ) changes from 10 to 40 and the number of attributes ( $n$ ) in an EMR is 10, 20, and 30, respectively. We can see that the time cost for the data buyer rises when  $\eta$  and  $n$  increase. From the figure, we can observe that when there are 40 acceptable hospitals and 30 attributes in an EMR, the computational cost for the data buyer is 82.37 ms. For the data seller, the computational overhead is mainly from the generation of zero-knowledge proof  $\pi_d$ , which is used to demonstrate that the data seller has the corresponding data that satisfy the requirements of the data buyer. The proof generation cost depends on the number of attributes ( $v$ ) in set  $I$ , which is disclosed for verifying the data availability, and the number of attributes ( $n$ ) in an EMR. Fig. 3(c) shows the computational cost for the data seller in the data trading phase. When  $v$  increases, the time cost decreases as there are fewer hidden values in the proof statement. If  $v$  is fixed, the computational cost for the data seller increases when  $n$  grows. From Fig. 3(c), we can see that when  $v = 15$  and  $n = 40$ , the time cost for the data seller is 44.67 ms.

We estimate the gas cost of storing data and performing cryptographic operations in the contract. Every opcode in the Ethereum Virtual Machine (EVM) specification has an associated gas fee [40]. For example, storing one word of data (32 bytes) costs 5000 gas for a zero value and 20000 gas for a nonzero

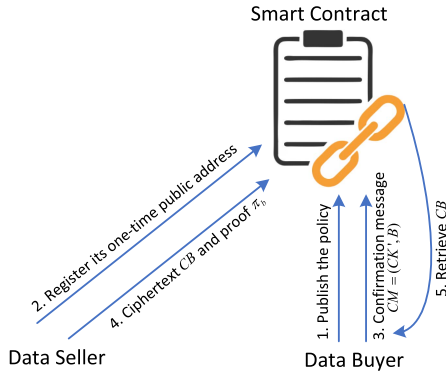


Fig. 4. Interaction between the trading participants and the smart contract.

TABLE VI  
COMPUTATIONAL AND STORAGE OVERHEAD FOR THE SMART CONTRACT

Function	Storage	Computation	Gas Cost
Policy Definition	$ Z_p  + 5 G_1  + 6 G_2  +  I  +  J $	NP	$70000 + 5000(n_I + n_J)$
Data Buyer Confirmation	$3 G_1 $	NP	120000
Ciphertext Verification	$5 G_1  + 4 Z_p $	$6M_1 + 8E_1$	328900

$|I| + |J|$  denotes the storage cost for sets  $I$  and  $J$ .  $n_I$  and  $n_J$  denote the number of elements in sets  $I$  and  $J$  respectively.

value. For the precompiled contract that implements a pairing check on the elliptic curve alt-bn128, the gas cost for the scalar multiplication on the elliptic curve is 6000, and point addition on the curve is 150.

As shown in Fig. 4, in the trading phase, the data buyer first uploads the defined policy on the smart contract, which includes the data requirements. Then, the data seller registers itself on the contract. If the off-chain verification of the data availability passes, the data buyer uploads a confirmation message ( $CM = (CK', B)$ ). After that, the data buyer generates ciphertext  $CB$  and zero-knowledge proof  $\pi_b$  and sends  $(CB, \pi_b)$  to the smart contract, which will verify whether  $\pi_b$  is valid. If  $\pi_b$  is verified, the data buyer can retrieve  $CB$  from the blockchain and obtain the symmetric encryption key of the trading data. In Table VI, we summarize the storage, computation, and gas cost for policy definition, data buyer confirmation, and ciphertext verification. For the bn128 curve,  $Z_p$  is 32 bytes,  $G_1$  is 64 bytes, and  $G_2$  is 128 bytes. We regard the size of an attribute in set  $I$  and set  $J$  as 8 bytes. From the table, we can see that after the data buyer uploads the defined policy, the gas cost for the data buyer to send confirmation message is about 120000, and it consumes about 328900 gas for validators to verify the zero-knowledge proof  $\pi_b$ .

## VIII. CONCLUSION

In this article, we have proposed a blockchain-based fair and fine-grained data trading scheme. In our scheme, data buyers can publish the data requirements, and a data seller can send only the required data to a data buyer without leaking other information. The security proof demonstrates that our scheme achieves fairness and privacy preservation. Beyond fair healthcare data

trading, the privacy-preservation and fine-grained trading strategy of our scheme can also be utilized to achieve flexible and practical data trading in other areas where data privacy is critical. For future work, considering that buyers may only require the data analysis results instead of obtaining the raw data, we aim to design a blockchain-based fair data trading scheme that allows a data buyer to attain the correct computation results without accessing the data of data sellers.

## REFERENCES

- [1] D. Zabelin, T. Yuyama, and T. Nakanishi, "The world is drowning in data. Why don't we trade it like on a stock exchange," 2022. [Online]. Available: <https://www.weforum.org/agenda/2022/01/data-trading-stock-exchange/>
- [2] Dawex, 2015. [Online]. Available: <https://www.dawex.com/en/>
- [3] F. Liang, W. Yu, D. An, Q. Yang, X. Fu, and W. Zhao, "A survey on Big Data market: Pricing, trading and protection," *IEEE Access*, vol. 6, pp. 15132–15154, 2018.
- [4] Xignite market data cloud platform on AWS, 2011. [Online]. Available: <https://aws.amazon.com/financial-services/partner-solutions/xignite-market-data-cloud-platform/>
- [5] X. Shen et al., "Data management for future wireless networks: Architecture, privacy preservation, and regulation," *IEEE Netw.*, vol. 35, no. 1, pp. 8–15, Jan./Feb. 2021.
- [6] L. Xue, D. Liu, J. Ni, X. Lin, and X. Shen, "Enabling regulatory compliance and enforcement in decentralized anonymous payment," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2022.3144991.
- [7] Y. Liu et al., "A blockchain-based decentralized, fair and authenticated information sharing scheme in zero trust Internet-of-Things," *IEEE Trans. Comput.*, vol. 72, no. 2, pp. 501–512, Feb. 2023.
- [8] D. Liu, C. Huang, J. Ni, X. Lin, and X. Shen, "Blockchain-cloud transparent data marketing: Consortium management and fairness," *IEEE Trans. Comput.*, vol. 71, no. 12, pp. 3322–3335, Dec. 2022.
- [9] S. Dziembowski, L. Ecekey, and S. Faust, "FairSwap: How to fairly exchange digital goods," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 967–984.
- [10] M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo, "Zero-knowledge contingent payments revisited: Attacks and payments for services," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 229–243.
- [11] Y. Li et al., "ZKCPlus: Optimized fair-exchange protocol supporting practical and flexible data exchange," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 3002–3021.
- [12] Y. J. Galletland and S. Wu, "Blockchain-based privacy-preserving fair data trading protocol," *Cryptol. ePrint Arch.*, 2021. [Online]. Available: <https://eprint.iacr.org/2021/1321>
- [13] J. Bobolz, F. Eidens, S. Krenn, S. Ramacher, and K. Samelin, "Issuer-hiding attribute-based credentials," in *Proc. Int. Conf. Cryptol. Netw. Secur.*, 2021, pp. 158–178.
- [14] O. Sanders, "Efficient redactable signature and application to anonymous credentials," in *Proc. IACR Int. Conf. Public-Key Cryptogr.*, 2020, pp. 628–656.
- [15] G. Maxwell, "The first successful zero-knowledge contingent payment," 2016. [Online]. Available: <https://bitcointalk.org/en/2016/02/26/zero-knowledge-contingent-payments-announcement/>
- [16] M. Petkus, "Why and how zk-SNARK works," 2019, *arXiv:1906.07221*.
- [17] S. Delgado-Segura, C. Pérez-Solà, G. Navarro-Arribas, and J. Herrera-Joancomartí, "A fair protocol for data trading based on bitcoin transactions," *Future Gener. Comput. Syst.*, vol. 107, pp. 832–840, 2020.
- [18] Y. Zhao, Y. Yu, Y. Li, G. Han, and X. Du, "Machine learning based privacy-preserving fair data trading in Big Data market," *Inf. Sci.*, vol. 478, pp. 449–460, 2019.
- [19] Y. Li, L. Li, Y. Zhao, N. Guizani, Y. Yu, and X. Du, "Toward decentralized fair data trading based on blockchain," *IEEE Netw.*, vol. 35, no. 1, pp. 304–310, Jan./Feb. 2021.
- [20] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, 2001.

- [21] S. F. Shahandashti and R. Safavi-Naini, "Threshold attribute-based signatures and their application to anonymous credential systems," in *Proc. Int. Conf. Cryptol. Afr.*, 2009, pp. 198–216.
- [22] J. Blömer and J. Bobolz, "Delegatable attribute-based anonymous credentials from dynamically malleable signatures," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2018, pp. 221–239.
- [23] P. Li, J. Lai, and Y. Wu, "Publicly traceable attribute-based anonymous authentication and its application to voting," *Secur. Commun. Netw.*, vol. 2021, 2021, pp. 1–17.
- [24] A. Islam et al., "NEWSTRADCOIN: A blockchain based privacy preserving secure news trading network," in *Proc. Int. Conf. Blockchain Technol.*, 2020, pp. 21–32.
- [25] W. Zou et al., "Smart contract development: Challenges and opportunities," *IEEE Trans. Softw. Eng.*, vol. 47, no. 10, pp. 2084–2106, Oct. 2021.
- [26] J. Liu and K. Ren, "Improving blockchains with client-assistance," *IEEE Trans. Comput.*, vol. 71, no. 5, pp. 1230–1236, May 2022.
- [27] R. Gay, D. Hofheinz, L. Kohl, and J. Pan, "More efficient (almost) tightly secure structure-preserving signatures," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2018, pp. 230–258.
- [28] J. Groth, "Efficient fully structure-preserving signatures for large messages," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2015, pp. 239–259.
- [29] J. Camenisch, M. Drijvers, and M. Dubovitskaya, "Practical UC-secure delegatable credentials with attributes and their application to blockchain," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 683–699.
- [30] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE Netw.*, vol. 35, no. 4, pp. 198–205, Jul./Aug. 2021.
- [31] Diabetes data set, 2019. [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/diabetes>
- [32] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen, "MuR-DPA: Top-down levelled multi-replica Merkle hash tree based secure public auditing for dynamic Big Data storage on cloud," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2609–2622, Sep. 2015.
- [33] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Int. Cryptol. Conf.*, 1991, pp. 129–140.
- [34] Y. Zhao, B. Niu, P. Li, and X. Fan, "A novel enhanced lightweight node for blockchain," in *Proc. Int. Conf. Blockchain Trustworthy Syst.*, 2020, pp. 137–149.
- [35] E. Palm, O. Schelén, and U. Bodin, "Selective blockchain transaction pruning and state derivability," in *Proc. Crypto Valley Conf. Blockchain Technol.*, 2018, pp. 31–40.
- [36] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 583–598.
- [37] X. Fan, B. Niu, and Z. Liu, "Scalable blockchain storage systems: Research progress and models," *Computing*, vol. 104, no. 6, pp. 1497–1524, 2022.
- [38] Miracl Library, 2006. [Online]. Available: <https://github.com/miracl/MIRACL>
- [39] G. C. Pereira, M. A. Simplício Jr, M. Naehrig, and P. S. Barreto, "A family of implementation-friendly BN elliptic curves," *J. Syst. Softw.*, vol. 84, no. 8, pp. 1319–1326, 2011.
- [40] K. Iyer and C. Dannen, *Building Games With Ethereum Smart Contracts*. New York, NY, USA: Apress, 2018.



**Liang Xue** (Member, IEEE) received the PhD degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2022. She is a postdoctoral research fellow with the School of Computer Science, University of Guelph. Her research interests include applied cryptography, cloud computing, and security and privacy on blockchain.



crowdsensing, and Internet of Things.

**Jianbing Ni** (Senior Member, IEEE) received the BE and MS degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2011 and 2014, respectively, and the PhD degree in electrical and computer engineering from the University of Waterloo, Waterloo, Canada, in 2018. He is currently an assistant professor with the Department of Electrical and Computer Engineering, Queen's University. His research interests include applied cryptography and network security, with current focus on cloud computing, smart grid, mobile



**Dongxiao Liu** (Member, IEEE) received the PhD degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2020. He is a postdoctoral research fellow with the Department of Electrical and Computer Engineering, University of Waterloo. His research interests include security and privacy in intelligent transportation systems, blockchain, and mobile networks.



**Xiaodong Lin** (Fellow, IEEE) received the PhD degree in information engineering from the Beijing University of Posts and Telecommunications, China, and the PhD degree (with Outstanding Achievement in Graduate Studies Award) in electrical and computer engineering from the University of Waterloo, Canada. He is currently a professor with the School of Computer Science, University of Guelph, Canada. His research interests include computer and network security, privacy protection, applied cryptography, computer forensics, and software security.



**Xuemin Shen** (Fellow, IEEE) is a University professor with the Department of Electrical and Computer Engineering, University of Waterloo. His research focuses on network resource management, wireless network security, AI for networks, and vehicular ad hoc networks. He is a Canadian Academy of engineering fellow, a Royal Society of Canada fellow, and a Chinese Academy of engineering foreign fellow. He received the R.A. Fessenden Award in 2019 from IEEE, Canada, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, and Joseph LoCicero Award in 2015 from the IEEE Communications Society. He is the president of IEEE Communications Society, and was the editor-in-chief of the *IEEE Internet of Things Journal*, and *IEEE Network*.