ZKDET: A Traceable and Privacy-Preserving Data Exchange Scheme based on Non-Fungible Token and Zero-Knowledge

Rui Song*, Shang Gao*, Yubo Song† and Bin Xiao*
*Department of Computing, The Hong Kong Polytechnic University, Hong Kong
†School of Cyber Science and Engineering, Southeast University, China
{csrsong, shanggao}@comp.polyu.edu.hk, songyubo@seu.edu.cn, csbxiao@comp.polyu.edu.hk

***Abstract*—With the advent of the Big Data era, industry, business and academia have developed various data exchange schemes to make data more economically beneficial. Unfortunately, most of the existing systems provide only one-time data exchanges without the ability to track the provenance and transformations of datasets. In addition, existing systems encrypt the data to protect data privacy, which hinders demanders from verifying the correctness of the data and evaluating its value.**

To provide data traceability and privacy while ensuring fairness during data exchanges, we design and implement ZKDET, a traceable data exchange scheme based on non-fungible token and zero-knowledge, which is able to (i) track all transformations of data during their lifecycle and record them on the blockchain; (ii) provide zero-knowledge proofs to securely guarantee that all complex transformations and data contents are correct and meet specific requirements; and (iii) warrant exchange fairness and data privacy in public storage platforms. Security analysis and evaluations on ZKDET show that it can support traceable data exchange while preserving data privacy and maintaining high throughput despite large data volumes.

***Index Terms*—Privacy-preserving, data exchange, blockchain, non-fungible token (NFT), zero-knowledge**

I. Introduction

Big Data leads a major era of change, transforming how people live and work. Governments, academic institutions and companies rely on Big Data to build more refined models for decision making and revenue creation. However, no entity can meet these needs by relying only on its own data. As a result, the data exchange industry has proliferated in recent years, providing marketplaces and coordination platforms for the supply and demand of data assets [1]. Nevertheless, the powerful authority to control data and transactions that such centralized marketplaces have can lead to many problems. For example, malicious behaviors of platforms or attacks against them may both lead to data leakage, which damages the interests of counterparties [2].

Studies have proposed replacing such centralized platforms with blockchain, whose advantages are apparent for data exchanges. First, as a decentralized system, blockchain can guarantee data correctness and validity through consensus mechanisms. Second, attacks against blockchain require colossal computing power, making the cost much higher than the potential benefit. Finally, some scaling efforts such as payment channels and other Layer 2 solutions can be utilized to increase throughput and reduce transaction fees, thereby shrinking the expense for data exchanges [3], [4].

Despite the introduction of blockchain, implementing a blockchain-based data exchange scheme remains a challenging endeavor, with several difficulties to be addressed:

- Data traceability is critical to data exchanges, since the value of a data asset is not only derived from the message it provides, but also endorsed by its provider [5]. Buyers may assess the authenticity and value of data assets by data provenance. However, unlike physical assets, data can be freely duplicated or redistributed, making data traceability particularly difficult.
- Datasets are subject to numerous transformations throughout their life cycles, such as *aggregation*, *partition*, *duplication* and *processing*, which are not adequately modeled and analyzed. In most existing schemes, data are packaged and sold as static datasets in their entirety. This deficiency not only hinders data traceability in the exchange process, but also restricts the exchange to static data while neglecting the scenarios of computational delegation and model trading.
- Existing privacy-preserving data exchange schemes cannot be adapted to public data exchange platforms, and only support one-time private data exchanges. They typically encrypt datasets and ensure exchange fairness through zero-knowledge-based protocols to prevent privacy breaches. However, they need to disclose the encryption key to finish the exchange, and thus cannot be used in the public platform.

To provide dynamic data traceability, privacy, and fair data exchanges, we propose ZKDET, a data exchange scheme based on non-fungible tokens (NFT) and zero-knowledge. This scheme enables free on-chain exchanges and traceability of universal datasets by providing proofs of transformations and exchanges in a zero-knowledge manner.

ZKDET uses NFTs as on-chain credentials for all exchanged data, which are stored in the distributed storage network. Taking advantage of NFT, ZKDET can uniquely identify data owners and track every change in their ownership. Benefiting from the thriving ecosystem of smart contracts and DeFi, data providers and demanders can exchange data tokens in ZKDET,

thereby monetizing and circulating the data flows.

ZKDET makes several significant modifications to the traditional data exchange paradigm to track all data transformations during their life cycle. Specifically, ZKDET supports *aggregation*, *partition*, *duplication* and *processing* of datasets. Given the dynamic nature of data susceptibility to mutations, it is essential to incorporate all the transformations to which they are subjected to provide data traceability. More importantly, supporting transformations enables data owners to perform data mining and model training based on existing datasets, and sell the computational results as new data assets. In this way, users can not just exchange data contents using ZKDET, but also delegate complicated computational tasks and pay for the efforts embedded.

To protect data privacy, ZKDET stores all datasets encrypted and keeps only their metadata in NFTs. However, data encryption hinders the verification of data correctness and validity, making it difficult for demanders to evaluate the value of targeted datasets. To address this issue, ZKDET introduces a commit-and-prove non-interactive zero-knowledge (CP-NIZK) scheme based on Plonk construct, which can perform data verification without revealing any message about data assets [6]. We further incorporate a circuit-friendly block cipher and a commitment primitive to reduce the proving and verification overhead [7], [8].

To summarize, the main contributions of this paper are as follows:

- We design and implement ZKDET, a blockchain-based data exchange scheme which provides data traceability and privacy while ensuring exchange fairness. ZKDET uses NFT as on-chain credentials for all data assets and provides full lifecycle traceability for data transformations and exchanges.
- We propose a generic data transformation protocol that provides proofs of transformations. These proofs ensure that all transformations satisfy specific predicates in a zero-knowledge manner, and provide traceability for all dynamic data assets.
- We propose a key-secure exchange protocol that enables exchange fairness for data assets in public storage. Given that all existing zk-based exchange protocols require key disclosure during interactions, this protocol fills the gap of exchange fairness under public data storage.
- We evaluate ZKDET in terms of both security and performance. The security properties of the above two protocols are rigorously demonstrated. ZKDET's performance is evaluated in terms of NIZK computational overhead and smart contract deployment cost.

The rest of the paper is organized as follows. Section II provides the necessary background and preliminaries. Section III delivers an overview of the general structure and workflow of ZKDET. Section IV details the two pivotal protocols in ZKDET, and section V analyzes their security properties. Section VI proposes the implementation of ZKDET and evaluates its performance. Section VII reviews some of the existing research in this area. The final section concludes the paper.

II. Preliminaries

A. Notation

We denote by $\lambda \in \mathbb{N}$ the security parameter and by 1^λ its unary representation. In this paper, we assume that all cryptographic algorithms take 1^λ as input, which is thus omitted from the input lists. If $\mathcal{S}$ is a finite set, we denote by $x \leftarrow \mathcal{S}$ the process of sampling x according to $\mathcal{S}$, and by $x \xleftarrow{R} \mathcal{S}$ a random and uniform one. For an integer or string s, we denote by $|s|$ its binary bit length. We denote by $[n]$ the set of integers $\{1, 2, \ldots, n\}$ and by $[0, n]$ the set $\{0, 1, \ldots, n\}$. We denote by $(d_i)_{i\in[\ell]}$ the tuple of elements $(d_1, d_2, \ldots, d_\ell)$.

We use letters in calligraphic font to denote parties in protocols, e.g., $\mathcal{S}$ for the seller and $\mathcal{B}$ for the buyer. We use letters in bold font to denote sets, and use letters in lowercase to denote elements in them, e.g., $\boldsymbol{D} = (d_i)_{i\in[n]}$.

B. Commitment scheme

Definition 2.1 (Commitment scheme): A commitment scheme for message $m \in \mathcal{M}$ is a tuple of 2 polynomial-time algorithms $\Gamma = (\mathsf{Commit}, \mathsf{Open})$ which work as follows:

- $(c, o) \xleftarrow{R} \mathsf{Commit}(m)$: a probabilistic algorithm which takes a message $m \in \mathcal{M}$ as inputs, outputs a commitment $c \in \mathcal{C}$ and an opening randomness (blinder) $o \in \mathcal{O}$;
- $v \leftarrow \mathsf{Open}(m, c, o)$: a deterministic algorithm which takes the commitment c, the randomness o and the original message m as inputs, outputs $v = 1$ when accepting the commitment or $v = 0$ when rejecting it.

A cryptographic secure commitment scheme is supposed to satisfy the properties of *hiding* and *binding* defined below.

Definition 2.2 (Binding): A scheme Γ satisfies the property of binding if

$$\Pr\begin{bmatrix} m_1 \neq m_2 \\ \mathsf{Open}(m_1, c, o_1) = \mathsf{Open}(m_2, c, o_2) = 1 \end{bmatrix} \leq \mathsf{negl}(\lambda)$$

holds for every efficient adversary $\mathcal{A}$ that output a tuple of 5 elements $(c; m_1, o_1; m_2, o_2)$.

Definition 2.3 (Hiding): A scheme Γ satisfies the property of hiding if

$$\Pr\begin{bmatrix} (c_1, o_1) \xleftarrow{R} \mathsf{Commit}(m_1) \\ r_1 \leftarrow \mathcal{A}(c_1) \end{bmatrix} : r_1 = 1 \approx \Pr\begin{bmatrix} (c_2, o_2) \xleftarrow{R} \mathsf{Commit}(m_2) \\ r_2 \leftarrow \mathcal{A}(c_2) \end{bmatrix} : r_2 = 1$$

holds for any two different messages $m_1, m_2 \in \mathcal{M}$.

C. Non-interactive zero knowledge (NIZK)

A Non-interactive zero-knowledge proof system is a zero-knowledge system in which the prover sends only one message to the verifier.

Definition 2.4 (NIZK): A NIZK for $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ is a tuple of 3 polynomial-time algorighms $\Pi = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ which work as follows:

- $(\mathsf{ek}, \mathsf{vk}) \xleftarrow{R} \mathsf{KeyGen}(1^\lambda, R)$: a probabilistic algorithm which takes a security parameter λ and a relation $R \in \mathcal{R}_\lambda$

as inputs, outputs a common reference string (crs) consisting of an evaluation key ek and a verification key vk;
- $\pi \xleftarrow{R} \mathsf{Prove}(\mathsf{ek}, x, w)$: a probabilistic algorithm which takes the evaluation key ek, a statement x and a witness w such that $R(x, w) = 1$, outputs a zero knowledge proof π;
- $b \leftarrow \mathsf{Verify}(\mathsf{vk}, x, \pi)$: a deterministic algorithm which takes the verification key vk, the statement x and the proof π as inputs, outputs $b = 1$ when accepting or $b = 0$ when rejecting.

A NIZK is supposed to satisfy the properties of *completeness*, *knowledge soundness* and *zero-knowledge* defined below.

Definition 2.5 (Completeness): A scheme Π is *complete* if

$$\Pr\left[\begin{matrix}(\mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda, R) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{ek}, x, w)\end{matrix} : \mathsf{Verify}(\mathsf{vk}, x, \pi) = 1\right] = 1$$

holds for any $\lambda \in \mathbb{N}$, $R \in \mathcal{R}$ and (x, w) s.t. $R(x, w) = 1$.

Definition 2.6 (Knowledge soundness): A scheme Π is *knowledge sound* if there exists an efficient *extractor* Ext s.t.

$$\Pr\left[\begin{matrix}\mathsf{crs} \leftarrow \mathsf{KeyGen}(1^\lambda, R) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ w \leftarrow \mathsf{Ext}(\mathsf{crs}, x, \pi)\end{matrix} : \begin{matrix}\mathsf{Verify}(\mathsf{vk}, x, \pi) = 1 \\ R(x, w) = 0\end{matrix}\right] \leq \mathsf{negl}(\lambda)$$

holds for every efficient adversary $\mathcal{A}$.

Definition 2.7 (Zero-knowledge): A scheme Π is *zero-knowledge* for relation generator $\mathcal{RG}$ if there exists a *simulator* $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{kg}}, \mathsf{Sim}_{\mathsf{prv}})$ s.t.

$$\Pr\left[\begin{matrix}(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda) \\ \mathsf{crs} \leftarrow \mathsf{KeyGen}(1^\lambda, R)\end{matrix} : \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R) = 1\right] \approx$$
$$\Pr\left[\begin{matrix}(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda) \\ (\mathsf{crs}, \mathsf{td}_\mathsf{k}) \leftarrow \mathsf{Sim}_{\mathsf{kg}}(R)\end{matrix} : \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R) = 1\right]$$

holds for all efficient adversary $\mathcal{A}$, and:

$$\Pr\left[\begin{matrix}(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda) \\ (\mathsf{crs}, \mathsf{td}_\mathsf{k}) \leftarrow \mathsf{Sim}_{\mathsf{kg}}(R) \\ (x, w, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{crs}, \mathsf{aux}_R) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{ek}, x, w)\end{matrix} : \begin{matrix}\mathcal{A}_2(\mathsf{st}, \pi) = 1 \\ R(x, w) = 1\end{matrix}\right] \approx$$
$$\Pr\left[\begin{matrix}(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda) \\ (\mathsf{crs}, \mathsf{td}_\mathsf{k}) \leftarrow \mathsf{Sim}_{\mathsf{kg}}(R) \\ (x, w, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{crs}, \mathsf{aux}_R) \\ \pi \leftarrow \mathsf{Sim}_{\mathsf{prv}}(\mathsf{crs}, \mathsf{td}_\mathsf{k}, x)\end{matrix} : \begin{matrix}\mathcal{A}_2(\mathsf{st}, \pi) = 1 \\ R(x, w) = 1\end{matrix}\right]$$

holds for any efficient adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Definition 2.8 (zkSNARK): A NIZK scheme Π is called zero-knowledge succinct non-interactive argument of knowledge (zkSNARK) if it satisfies Definition 2.4 and the property of succinctness, i.e., the running time of $\mathsf{Verify}(\mathsf{vk}, x, \pi)$ is $\mathcal{O}(\lambda, |x|, \log|w|)$ and the proof size is $\mathcal{O}(\lambda, \log|w|)$.

D. Non-fungible tokens

Non-fungible token (NFT) is a type of token derived from smart contracts. In contrast to fungible tokens standardized by ERC-20, NFTs are unique, indivisible and not directly interchangeable on-chain credentials. NFTs cannot be divided into smaller units while only existing as a whole.

The Ethereum ERC-721 specification standardizes the transfer, recording and tracking operations of NFTs, providing a uniform interface and data structure [9]. Each NFT based on the ERC-721 standard is identified by a unique `tokenId` which cannot be changed within the underlying smart contract.

As digital credentials stored on distributed ledgers, NFTs can theoretically be associated with any easily reproducible objects such as videos, images, digital art, and other digital assets with originality, and serve as proof of ownership and certificates of authenticity. By using NFTs on smart contracts, anyone can easily prove the existence and attribution of digital assets. In addition, any successful transaction on the NFT marketplace gives the creator a financial reward. Full historical tradability, high liquidity and convenient interoperability make NFT a promising intellectual property solution for digital assets.

III. zkDET overview

In this section, we will first outline the basic architecture of zkDET by delivering a strawman design, as shown in Figure 1. After that, we will discuss several technical challenges along with potential solutions to address them.

A. Binding data to non-fungible tokens

As mentioned earlier, NFTs are ideally suited as on-chain credentials for digital assets and proofs of their ownership. Any data owner can package their data holdings and mint them into NFTs. Using the infrastructure provided by the ERC-721 specification, data owners can freely trade tokens in the on-chain marketplace, thereby enabling data circulation and monetization.

To achieve this, a data owner first needs to publish its data asset $\boldsymbol{D}$ to a decentralized storage network like IPFS, obtaining the URI to access the data. To ensure data privacy, the owner needs to encrypt $\boldsymbol{D}$ by $\hat{\boldsymbol{D}} \leftarrow \mathsf{Enc}(k, \boldsymbol{D})$ before uploading it. Note that content addressing in IPFS is based on the hash digest of datasets, we can thus treat the data's URI as its hash commitment, i.e., $\mathsf{URI} := c \leftarrow H(\hat{\boldsymbol{D}})$. Anyone in the network can request an encrypted copy of $\boldsymbol{D}$ via the Distributed Hash Table (DHT) using the URI.

The data owner then records the URI into the smart contract and mints an NFT, which is the unique credential of $\boldsymbol{D}$ on the blockchain. ERC-721 specifies that each NFT has a field named `tokenId` as its unique identifier in the very contract. Thus given a `tokenId` in a smart contract, anyone can index the corresponding NFT and address $\hat{\boldsymbol{D}}$ in the storage network.

B. Data transformation and traceability

Regardless of the type or content of tokens, on-chain operations can be broadly summarized as follows:

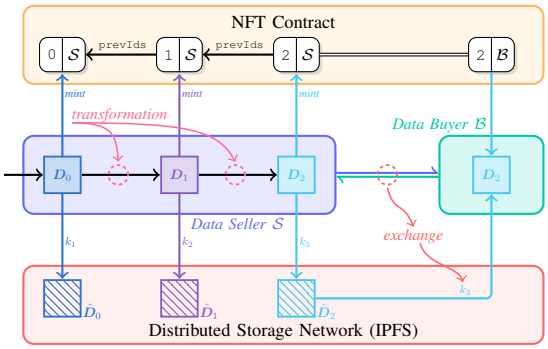

Fig. 1. The basic workflow and structure of ZKDET.

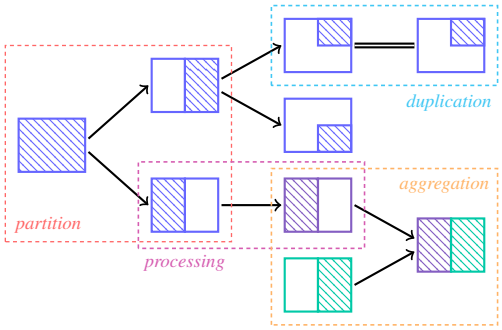

Fig. 2. Fundamental formulae of data transformation in ZKDET.

1) *Minting*: publishing a new token to the blockchain via smart contract, which puts the data asset into circulation;
2) *Transferring*: shifting the ownership of a token from one address to another, indicating the exchange of ownership of a data asset;
3) *Burning*: destroying an existing token in the contract, taking the corresponding data asset out of circulation.

Nevertheless, unlike other physical or virtual assets, datasets can be transformed or processed, as we have analyzed. Most existing data exchange schemes fail to consider this and focus only on one-time transactions where the ownership of data assets is simply transferred from one entity to another. In ZKDET, however, the recording of transformations provides traceability to data assets, which helps buyers track data provenance and mutations, and thus assess the value of the data. In a nutshell, all data transformations are abstracted into the following formulae:

4) *Aggregation*: merging multiple data assets into a new one and minting a token corresponding to it;
5) *Partition*: splitting a data asset into several ones and minting tokens corresponding to them;
6) *Duplication*: replicating the contents of a data asset and minting a new token corresponding to the replica;
7) *Processing*: Mutating a data asset by computing or calculating, and minting a new token corresponding to the derived one.

It is apparent that any transformation in the form or content of datasets can be achieved by combining the above operations. To demonstrate the relationship between data assets before and after transformations, each NFT incorporates a field called `prevIds[]` which records the `tokenId` of all its parent tokens. Figure 2 illustrates an example where some data assets undergo multiple transformations, which can be traced through `prevIds[]` up to their sources.

Although the inheritance relations of tokens are recorded on the chain, the corresponding datasets are not guaranteed to conform to the claim. A malicious $\mathcal{P}^*$ can upload a false dataset $\tilde{\boldsymbol{D}}$ and claim that it is derived from an existing dataset $\hat{\boldsymbol{S}}$ under transformation f. Data encryption makes this attack even easier. Since a third party does not hold the decryption keys k_1 and k_2, there is no way for it to obtain the plaintexts $\boldsymbol{S}$ and $\boldsymbol{D}$ and check whether $\boldsymbol{D} = f(\boldsymbol{S})$ holds. To solve this problem, the data owner $\mathcal{P}$ needs to prove in zero-knowledge that:

1) She knows 2 datasets $\boldsymbol{S} = (s_i)_{i\in[n]}$ and $\boldsymbol{D} = (d_j)_{j\in[m]}$ which are the plaintexts corresponding to the ciphertexts $\hat{\boldsymbol{S}} = (\hat{s}_i)_{i\in[n]}$ and $\hat{\boldsymbol{D}} = (\hat{d}_j)_{j\in[m]}$, respectively, i.e.,
$$\bigwedge_{i\in[n]} \hat{s}_i = \mathsf{Enc}(k_1, s_i) \ \wedge \ \bigwedge_{j\in[m]} \hat{d}_j = \mathsf{Enc}(k_2, d_j) \quad (1)$$
2) She knows the keys k_1 and k_2 used for encryption, which are committed to c_1 and c_2 with randomness o_1 and o_2, respectively, i.e.,
$$\mathsf{Open}(k_1, c_1, o_1) = 1 \ \wedge \ \mathsf{Open}(k_2, c_2, o_2) = 1 \quad (2)$$
3) $\boldsymbol{D}$ is obtained from $\boldsymbol{S}$ via f, i.e.,
$$\boldsymbol{D} = f(\boldsymbol{S}) \quad (3)$$

Given that both $\hat{\boldsymbol{S}}$ and $\hat{\boldsymbol{D}}$ can be publicly accessed, $\mathcal{P}$ publishes the commitments c_1 and c_2 to the token along with a proof π_f by:
$$\pi_f \xleftarrow{\mathrm{R}} \mathsf{Prove}(\mathsf{ek}, (\hat{\boldsymbol{S}}, \hat{\boldsymbol{D}}, c_1, c_2), (\boldsymbol{S}, \boldsymbol{D}, k_1, k_2, o_1, o_2, \mathsf{aux}))$$
where aux is an auxiliary witness. She also needs to prove that $\boldsymbol{D}$ is well-formed in a proof π_ℓ as its format matches the data structure declared in the NFT metadata. In this way, anyone can verify this transformation using (π_f, π_ℓ) without knowing any other message about $\boldsymbol{S}$ and $\boldsymbol{D}$.

C. Data exchange with fairness and privacy

As a data exchange scheme, exchange fairness and data privacy are primary objectives. There have been many discussions on fair data exchange, where the current de facto standard is the Zero-Knowledge Contingent Payment (ZKCP) protocol [10]. The ZKCP protocol is able to conduct fair data exchanges while relying on no trusted third parties. Instead, it uses Bitcoin scripts or smart contracts as arbiters, allowing sellers to prove to buyers that the exchanged data $\boldsymbol{D}$ satisfies a public predicate ϕ s.t. $\phi(\boldsymbol{D}) = 1$ using NIZK.

Consider an interaction of data exchange using ZKCP with three counterparties, i.e., a seller $\mathcal{S}$, a buyer $\mathcal{B}$ and an arbiter $\mathcal{J}$ implemented by a smart contract. Before the interaction, $\mathcal{S}$ launches a clock auction which locks its token for sale,

corresponding to an encrypted dataset $\hat{\boldsymbol{D}}$. $\mathcal{S}$ also attaches a predicate ϕ of the plaintext $\boldsymbol{D}$ to the auction contract for all potential buyers. The interaction is as follows:

1) *Deliver*: $\mathcal{S}$ is initialized by $(\boldsymbol{D}, k, \hat{\boldsymbol{D}}, \phi)$, where k is the encryption key. She computes the hash of k by $h \leftarrow H(k)$ and generates a proof of knowledge $\pi_p \leftarrow \mathsf{Prove}(\mathsf{ek}, (\hat{\boldsymbol{D}}, h), (\boldsymbol{D}, k))$, which indicates the relation:
$$\phi(\boldsymbol{D}) = 1 \wedge \hat{\boldsymbol{D}} = \mathsf{Enc}(k, \boldsymbol{D}) \wedge h = H(k)$$
holds. She sends the tuple (h, π_p) to $\mathcal{B}$.
2) *Verify*: $\mathcal{B}$ is initialized by $(\hat{\boldsymbol{D}}, \phi)$. Upon receiving $(\hat{\boldsymbol{D}}, h, \pi_p)$, $\mathcal{B}$ verifies the proof π_p by:
$$b \leftarrow \mathsf{Verify}(\mathsf{vk}, (\hat{\boldsymbol{D}}, h), \pi_p)$$
If $b = 1$, $\mathcal{B}$ submits a payment to $\mathcal{J}$ along with the h she received. By doing this, she claims that anyone who can provide a valid pre-image of h can redeem the payment locked in the contract.
3) *Open*: $\mathcal{S}$ checks that h in $\mathcal{J}$ is correct and the payment locked is as previously agreed. She discloses k to $\mathcal{J}$.
4) *Finalize*: $\mathcal{J}$ verifies whether $h = H(k)$ holds and forwards the payment to $\mathcal{S}$ if it is the case. Meanwhile, $\mathcal{B}$ gets k from $\mathcal{J}$ and decrypts $\hat{\boldsymbol{D}}$ by $\boldsymbol{D} \leftarrow \mathsf{Dec}(k, \hat{\boldsymbol{D}})$.

It has been proved that any malicious buyer $\mathcal{B}^*$ cannot obtain any information about $\boldsymbol{D}$ other than what is provided by ϕ without completing the payment. Also, no malicious seller $\mathcal{S}^*$ can obtain payment by providing $\tilde{\boldsymbol{D}}$ which contradicts ϕ. This ensures fairness and data privacy in data exchanges [10].

D. Issues and challenges

So far, we have introduced the main components of ZKDET through a strawman design. However, this design has many problems and thus cannot be applied in the real world. To solve these problems, the following challenges should be addressed.

*1) **Challenge 1**: How to design proof predicates and arithmetic circuits for complicated transformations?*

To prove the correlations of datasets during transformations, predicates should be designed for NIZK proofs. However, it is impossible to enumerate all potential operations for practical scenarios. Nevertheless, we implement a library of fundamental cryptographic and mathematical gadgets to construct predicates for complicated relations, see IV-D. In addition, we illustrate how to combine these gadgets to provide proofs for data processing applications with several examples in IV-E.

*2) **Challenge 2**: How to improve the efficiency of zero-knowledge proofs in large-scale data scenarios?*

Two factors significantly restrict the efficiency and throughput of ZKDET. One is that the proofs of encryption in π_t are redundant for continuous data transformations; the other is that existing NIZK schemes are not suitable for large data volumes, especially when encryption is needed. To reduce proving time, we introduce MiMC cipher and Poseidon hash, which can effectively reduce the number of constraints in arithmetic circuits while obtaining security equivalent to that of traditional cryptographic primitives, see IV-B and IV-C.

*3) **Challenge 3**: How to preserve encryption key while ensuring exchange fairness?*

A critical flaw of the ZKCP-based exchange protocol introduced in III-C is that the plaintext $\boldsymbol{D}$ can be accessed by anyone once the exchange is finished, since $\mathcal{S}$ is obliged to disclose k to $\mathcal{J}$ in the *Open* phase. Given that $\hat{\boldsymbol{D}}$ is publicly stored, anyone can get k and decrypt it. A key insight of ZKDET is to conceal k from $\mathcal{J}$ and only provide a proof π_k which justifies k and provides exchange fairness. We will detail a two-phase protocol in IV-F to catch the above idea.

IV. Protocols detail

In this section, we will first present the threat model of ZKDET. Subsequently, we will elaborate on a *generic data exchange protocol* which provides predicates for data transformations. Finally, we will introduce a *key-secure exchange protocol* which conceals encryption keys while guaranteeing exchange fairness and data privacy.

A. Threat model and security properties

We assume that participants in ZKDET are all semi-honest, i.e., they will follow the execution of the protocol as they interact, but keep all intermediate computational states of the protocol. We consider a static, malicious adversary $\mathcal{A}$. Any compromised entity can deviate from the protocol at will and reveal its state to the adversary $\mathcal{A}$.

Blockchain. ZKDET's assumptions about blockchain systems are no different from standard assumptions, i.e., the blockchain is tamper-resistant and consistent. Tamper-resistance means that once a transaction has been credited to the blockchain and confirmed by a certain number of subsequent blocks, it cannot be rolled back. Consistency means that after a certain period of time, the blockchain system presents a consistent state to the outside world, and all users have the same view of its state. Apart from that, we assume that all users have access to all transactions in the blockchain.

Network and Storage. We assume that the network communication between entities does not leak the content to third parties. In addition, given corresponding URIs, all datasets in the distributed storage are publicly available. Any persisted dataset will not be removed unless explicitly requested by its owner. Any tampering with dataset will result in change in its digest and URI, and thus cannot be concealed.

B. Generic data transformation protocol

We introduced in III-B a naive protocol that implements on-chain recording and tracking of data transformations. However, it suffers from a significant problem. For each transformation, π_t contains two proofs of encryption for datasets before and after mutation, as shown in Equation 1. While in continuous transformations, these proofs are repeatedly included in different π_t, resulting in redundancy. To better illustrate this issue, suppose a dataset $\boldsymbol{D}_2$ is obtained from $\boldsymbol{D}_1$ by f_1, and subsequently transformed into $\boldsymbol{D}_3$ under f_2. The owner $\mathcal{P}$ needs to compute π_{t_1} and π_{t_2} for f_1 and f_2 respectively, both of which contain the proof of $\hat{\boldsymbol{D}}_2 = \mathsf{Enc}(k_2, \boldsymbol{D}_2)$.

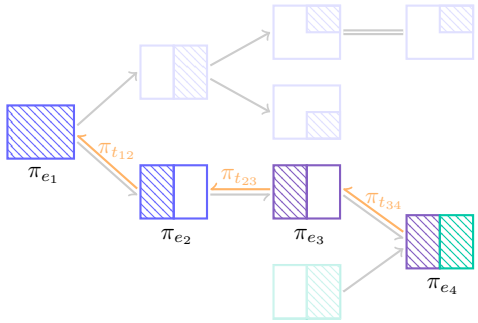

Fig. 3. Chaining up proofs of transformation π_t.

A critical insight is that the proofs of encryption can be decoupled from those of transformation, which can be reused in subsequent transformations. Suppose a data owner $\mathcal{P}$ owns a dataset $\boldsymbol{S}$ and wants to transform it into a new $\boldsymbol{D}$. To this end, $\mathcal{P}$ needs to perform the following operations:

1) She commits to $\boldsymbol{S}$ by $(c_s, o_s) \xleftarrow{\mathrm{R}} \mathsf{Commit}(\boldsymbol{S})$ and proves that she knows a key k_s that encrypts $\boldsymbol{S}$ to $\hat{\boldsymbol{S}}$, i.e., $\bigwedge_{i\in[n]} \hat{s}_i = \mathsf{Enc}(k_s, s_i)$ by:
$$\pi_{e_s} \xleftarrow{\mathrm{R}} \mathsf{Prove}(\mathsf{ek}, (\hat{\boldsymbol{S}}, c_s), (\boldsymbol{S}, k_s, o_s, \mathsf{aux}))$$
2) She transforms $\boldsymbol{S}$ into $\boldsymbol{D}$ using f and commits to $\boldsymbol{D}$ by $(c_d, o_d) \xleftarrow{\mathrm{R}} \mathsf{Commit}(\boldsymbol{D})$. She then proves the relation $\boldsymbol{D} = f(\boldsymbol{S})$ by:
$$\pi_t \xleftarrow{\mathrm{R}} \mathsf{Prove}(\mathsf{ek}, (c_s, c_d), (\boldsymbol{S}, \boldsymbol{D}, o_s, o_d, \mathsf{aux}))$$
3) She randomly chooses a key $k_d \xleftarrow{\mathrm{R}} \mathcal{K}$ and encrypts $\boldsymbol{D}$ by $\hat{d}_j \leftarrow \mathsf{Enc}(k_d, d_j)$ for $j \in [m]$. Similar to 1), she generates a proof of encryption for $\hat{\boldsymbol{D}}$ by:
$$\pi_{e_d} \xleftarrow{\mathrm{R}} \mathsf{Prove}(\mathsf{ek}, (\hat{\boldsymbol{D}}, c_d), (\boldsymbol{D}, k_d, o_d, \mathsf{aux}))$$

In this way, we split π_f in III-B into a *proof of transformation* π_t and two *proofs of encryption* π_{e_s}, π_{e_d}, which can be reused later. Imagine that $\mathcal{P}$ continues to transform $\boldsymbol{D}$ to a new dataset $\boldsymbol{M}$ afterward, she does not need to compute π_{e_d} once again, which halves the cost of proof generation. Another benefit of proof decoupling is that proofs of transformation π_t can form a *proof chain* that provides continuous validation from data sources, as shown in Figure 3. All statements required for proof validation are publicly available, which empowers participants to evaluate datasets throughout their lifecycle.

Note that the above proofs are all based on the commitment of datasets, where the commit-and-prove NIZK (CP-NIZK) scheme can be utilized to enable the conjunction of relations with shared inputs [11]. According to the composition property of CP-NIZK, the combination of the above three separate proofs are equivalent to π_t.

C. Optimization for encryption and commitment

A critical problem of the above protocol is that providing NIZK proofs for encryption and commitment is time-consuming and arithmetic-intensive. The circuits constructed for AES and SHA-256 contain a tremendous number of non-linear operations, making the number of constraints unacceptable. Even if the circuits are optimized, an implementation of AES for about 1000 blocks still contain millions of constraints, which is completely impractical [12].

1) MiMC cipher: To reduce the circuit complexity, MiMC cipher is used as the encryption primitive, which is circuit-friendly in NIZK [7]. It requires only 82 multiplications in $\mathbb{F}_p$ to guarantee 129 bits security. In ZKDET, we use MiMC-p/p in counter (CTR) mode for data encryption. Entry d_i in dataset $\boldsymbol{D}$ is firstly encoded to $\bar{d}_i \in \mathbb{F}_p$ and encrypted by:
$$\hat{d}_i \leftarrow \bar{d}_i + \mathsf{MiMC}(k, \mathsf{nonce} + i)$$
where k is the encryption key and $\mathsf{nonce} \in \mathbb{F}_p$ is a randomly chosen field element. Taking advantage of multi-core architecture and pipeline construct, MiMC-CTR can be optimized for parallel computing and significantly improve proof efficiency.

2) Poseidon hash: Similar to MiMC, Poseidon is a cryptographic hash scheme optimized for arithmetic circuits. It is based on a strategy called *substitution-permutation* networks and *partial* rounds structure. For messages with the same bit length, the number of constraints by Poseidon is only about one-eighth that of Pedersen commitment [8]. For a permutation with width w and R rounds, optimized Plonk initializes only $(w+11)R$ point productions with proof containing $w+3$ elements in $\mathbb{G}$ and $2w$ elements in $\mathrm{GF}(p)$. This optimization further brings about a 25-40 times performance improvement.

D. Predicates for transformations

A generic data transformation protocol is introduced in IV-B without clarifying the generation of π_t for a specific formula f. Predicates are developed for fundamental formulae classified in III-B, along with corresponding arithmetic circuits.

1) *Duplication*: the data owner $\mathcal{P}$ replicates an existing dataset $\boldsymbol{S} = (s_i)_{i\in[n]}$ to get $\boldsymbol{D} = (d_j)_{j\in[m]}$, and proves that the content of both is identical, i.e.,
$$n = m \wedge \bigwedge_{i\in[n]} d_i = s_i$$
The number of constraints is of $\mathcal{O}(n)$ level and contains no additive or multiplicative calculations on $\mathbb{F}_p$.
2) *Aggregation*: $\mathcal{P}$ combines x existing datasets $(\boldsymbol{S}_k)_{k\in[x]}$ into a derived $\boldsymbol{D} \in \mathbb{F}_p^m$. Suppose that $\boldsymbol{S}_k$ is converged into $\boldsymbol{D}$ in order of k. Let $\boldsymbol{S}_k = (s_{kj})_{j\in[n_k]} \in \mathbb{F}_p^{n_k}$, $\mathcal{P}$ needs to prove that $\boldsymbol{D}$ contains all elements in $(\boldsymbol{S}_k)_{k\in[x]}$, i.e.,
$$m = \sum_{k=1}^{x} n_k \wedge \bigwedge_{i\in[x]} \left(\bigwedge_{j\in[n_i]} s_{ij} = d_{\sum_{k=1}^{i-1} n_k + j} \right)$$
3) *Partition*: $\mathcal{P}$ splits $\boldsymbol{S}$ into y derived $(\boldsymbol{D}_k)_{k\in[y]}$. She needs to prove that the split is exhaustive and mutually exclusive with respect to $\boldsymbol{S}$, i.e.,
$$\bigwedge_{k\in[y]} n_k \neq 0 \wedge \boldsymbol{S} = \bigcup_{k\in[y]} \boldsymbol{D}_k \wedge \bigwedge_{\substack{i,j\in[y] \\ i\neq j}} \boldsymbol{D}_i \cap \boldsymbol{D}_j = \varnothing$$

4) *Processing*: Different from transformations above, there is neither a fixed paradigm nor an exhaustive list for data processing. Nevertheless, a gadget library is designed for common calculations such as:
   - *Mathematical primitives*: algebraic and matrix operation, logarithmic computation, linearization, etc.;
   - *Cryptographic primitives*: encryption, hashing, elliptic curves and pairing, Merkle proof, etc.

   Also, we will present below several examples on how to utilize these gadgets to provide predicates for complicated data processing operations.

E. Applications in data processing

zkDET allows data owners to train models based on datasets and sell them as data assets. This allows participants to not only exchange data contents but also to delegate computational tasks in the marketplace. To present a proof of concept, we use two examples below to show how to combine gadgets and design predicates for model training scenarios.

1) Proof for logistic regression: We begin with logistic regression analysis, which is a commonly used statistical model. Suppose that the entries in the source $\boldsymbol{S}$ are a series of points $[(x_{ij})_{j\in[k]}, y_i]$ and the derived $\boldsymbol{D} \in \mathbb{F}_p^k$ is the parameters $(\hat{\beta}_j)_{j\in[0,k]}$ of predictor function $\hat{y} = \hat{\beta}_0 + \sum_{j\in[k]} \hat{\beta}_j \hat{x}_j$. Then the loss function is:

$$J(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n} [y_i \log h_{\boldsymbol{\beta}}(x_i) + (1-y_i)\log(1-h_{\boldsymbol{\beta}}(x_i))]$$

where $h_{\boldsymbol{\beta}}(x) = [1+\exp(-\boldsymbol{\beta}^{\mathrm{T}}x)]^{-1}$ and $\boldsymbol{\beta} = (\beta_j)_{j\in[0,k]}$. To verify $\boldsymbol{D}$, $\mathcal{P}$ needs to prove that the difference of J between the last two iterations is less than a certain threshold ϵ, i.e.,

$$||J(\boldsymbol{\beta}^{(k+1)}) - J(\boldsymbol{\beta}^{(k)})|| \leq \epsilon$$

Since $\boldsymbol{D}$ contains only $\boldsymbol{\beta}^{(k)}$ for the k-th iteration, $\boldsymbol{\beta}^{(k+1)}$ can be derived from $\boldsymbol{\beta}^{(k)}$ in the proof by:

$$\beta_i^{(k+1)} = \beta_i^{(k)} - \alpha x_i [h_\beta(x_i) - y_i]$$

where α is the step size. Although multiple iterations are required to compute $\boldsymbol{\beta}$ using gradient descent, proving the correctness of $\boldsymbol{D}$ requires only the last two iterations and is thus computationally acceptable. Note that when constructing circuits, nonlinear operations like exponential and logarithmic need to be evaluated, where our gadget library can be of help.

2) Proof for transformers: zkDET is also promising in the field of deep learning. Take the recently popular transformer as an example. The transformer model consists of encoder and decoder structures, which are composed of attention construct and feed-forward neural network, respectively.

Multi-Head Attention consists of multiple scaled dot-product attention constructs, where the input $\boldsymbol{S} = (s_i)_{i\in[n]}$ is transformed into queries q_i, keys k_i and values v_i by inner product, such as $q_i = s_i \cdot W_Q$. The output is:

$$z_i = \mathrm{softmax}(\frac{q_i \cdot k_i^{\mathrm{T}}}{\sqrt{d_k}}) \cdot v_i$$

Feed-Forward Neural Network consists of two fully connected layers where the activation function of the first layer is ReLU and the second layer has no activation function. For the output z_i of attention, the corresponding output is:

$$d_i = \max(0, z_i \cdot W_1 + b_1) \cdot W_2 + b_2$$

Then the derived dataset is $\boldsymbol{D} = (d_i)_{i\in[m]}$. Although model training may involve multiple rounds of iterations, proving the correctness of the model is a one-time task that can be done in $O(n)$ time. Moreover, the proof of decoders can be parallelized using *Teaching Forcing* to further optimize the performance.

F. Key-secure data exchange protocol

ZKCP protocol is introduced for data exchange in III-C with a significant vulnerability. It discloses the key k in the *Open* phase, which can be acquired by any third party thereafter to decrypt the publicly stored $\hat{\boldsymbol{D}}$. To address this issue, we develop a key-secure two-phase data exchange protocol, where *key-secure* indicates that k will never be disclosed but only passed between $\mathcal{S}$ and $\mathcal{B}$. It is called a *two-phase* protocol because there are two phases in the interaction where $\boldsymbol{D}$ and k are delivered and verified, respectively.

As in III-C, the seller $\mathcal{S}$ is initialized by $(\boldsymbol{D}, k, \hat{\boldsymbol{D}}, \phi)$ s.t. $\phi(\boldsymbol{D}) = 1$ and $\hat{\boldsymbol{D}} = \mathsf{Enc}(k, \boldsymbol{D})$, the buyer $\mathcal{B}$ is initialized by $(\hat{\boldsymbol{D}}, \phi)$, and the arbiter $\mathcal{J}$ is initialized with the commitment c of k. The interaction between $\mathcal{S}$ and $\mathcal{B}$ is as follows:

1) Data validation phase

- $\mathcal{S}$ commits to $\boldsymbol{D}$ by $(c_d, o_d) \xleftarrow{\mathrm{R}} \mathsf{Commit}(\boldsymbol{D})$ and generates a proof $\pi_p \xleftarrow{\mathrm{R}} \mathsf{Prove}(\mathsf{ek}, (\hat{\boldsymbol{D}}, c_d, \phi), (\boldsymbol{D}, k, o_d, \mathsf{aux}))$ which indicates:

  $$\phi(\boldsymbol{D}) = 1 \wedge \hat{\boldsymbol{D}} = \mathsf{Enc}(k, \boldsymbol{D}) \wedge \mathsf{Open}(\boldsymbol{D}, c_d, o_d) = 1$$

  holds. She sends the tuple (c_d, π_p) to $\mathcal{B}$.
- $\mathcal{B}$ verifies π_p by $b \leftarrow \mathsf{Verify}(\mathsf{vk}, (\hat{\boldsymbol{D}}, c_d, \phi), \pi_p)$. If $b = 1$, she randomly chooses a key $k_v \xleftarrow{\mathrm{R}} \mathcal{K}$ and sends it back to $\mathcal{S}$. Meanwhile, she submits a payment to $\mathcal{J}$ along with a hash h_v by $h_v \leftarrow H(k_v)$.

2) Key negotiation phase

- $\mathcal{S}$ derives a key k_c by $k_c \leftarrow k + k_v$ and generates a proof $\pi_k \xleftarrow{\mathrm{R}} \mathsf{Prove}(\mathsf{ek}, (k_c, c, h_v), (k_v, k, o, \mathsf{aux}))$ indicating:

  $$\mathsf{Open}(k, c, o) = 1 \wedge h_v = H(k_v) \wedge k_c = k + k_v$$

  holds. She sends (k_c, π_k) to $\mathcal{J}$.
- $\mathcal{J}$ verifies π_k by $b \leftarrow \mathsf{Verify}(\mathsf{vk}, (k_c, c, h_v), \pi_k)$. It forwards the payment to $\mathcal{S}$ if $b = 1$, otherwise returns it to $\mathcal{B}$.
- $\mathcal{B}$ derives k by $k \leftarrow k_c - k_v$ and decrypts $\hat{\boldsymbol{D}}$ by $\boldsymbol{D} \leftarrow \mathsf{Dec}(k, \hat{\boldsymbol{D}})$.

The interaction is shown in Figure 4. Based on this protocol, buyers and sellers only need to pass the key outside the blockchain and verify it via $\mathcal{J}$. Note that in *data verification phase*, we do not commit to k as in ZKCP but the plaintext $\boldsymbol{D}$ to i) introduce arguments modularity and relation composition using the CP-NIZK scheme; and ii) reuse the proofs π_e and π_f generated in the data transformation protocol.

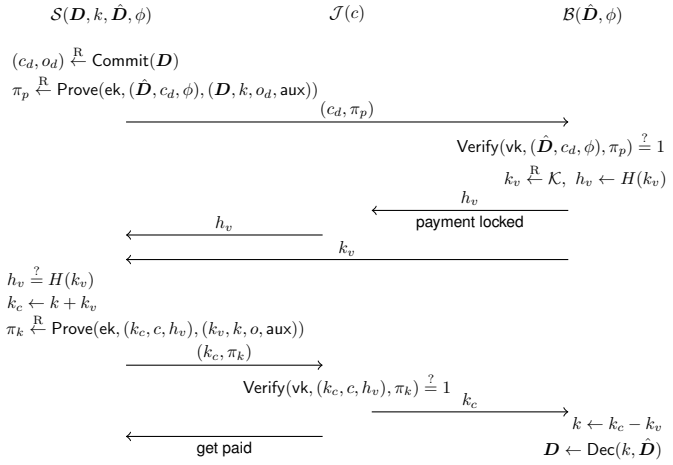

Fig. 4. Key-secure data exchange protocol.

Notice that the relation proved by π_p has two parts, i.e., i) $\hat{\boldsymbol{D}}$ comes from the encryption of $\boldsymbol{D}$; and ii) $\boldsymbol{D}$ satisfies the predicate ϕ. A key insight is that the argument for i) can be provided by exactly π_e and thus does not need to be computed again during the interaction, while the argument for ii) can be provided by proofs of f in the vast majority of cases. This is because, for any $\boldsymbol{D}$, complete knowledge can be learnt through its data sources and undergoing transformations. While the latter can be provided by chains of π_f in IV-B, and the former can be produced by decentralized oracles like DECO [13].

V. Security Analysis

In this section, we prove that the generic transformation protocol and the two-phase data exchange protocol introduced in Section IV are secure.

A. Security of transformation protocol

Regarding the generic data transformation protocol, we have the following theorem.

Theorem 5.1: Given that the commitment scheme Γ satisfies definition 2.2 and 2.3, and the NIZK scheme Π satisfies definition 2.5, 2.6 and 2.7, the generic data transformation protocol in IV-B has the following security properties:

- *Integrity*: no potential malicious prover $\mathcal{P}^*$ can falsify a source dataset $\hat{\boldsymbol{S}}$ or make the verifier $\mathcal{V}$ accept invalid statements about f. That is, except for a negligible probability, $\mathcal{P}^*$ cannot convince $\mathcal{V}$ to accept any of the false π_{e_s}, π_{e_d} or π_t.
- *Privacy*: no potential malicious verifier $\mathcal{V}^*$ who knows only public statements can learn any knowledge about datasets before and after transformation f ($\boldsymbol{S}$ and $\boldsymbol{D}$) other than what can be directly inferred from f.

1) Proof of integrity: Given the computational knowledge soundness of Π, except for a negligible probability $\mathsf{negl}(\lambda)$, no potential malicious prover $\mathcal{P}^*$ can generate a false π_t to convince the verifier $\mathcal{V}$ to accept $\boldsymbol{D} = f(\boldsymbol{S})$.

On the other hand, if $\mathcal{V}$ accepts π_{e_s} at the end of the interaction, there must exist an extractor Ext which, except for a negligible probability $\mathsf{negl}(\lambda)$, is able to output a tuple $(\boldsymbol{S}', k_s', o_s')$ s.t.

$$\bigwedge_{i\in[n]} \hat{s}_i = \mathsf{Enc}(k_s', s_i') \ \wedge\ \mathsf{Open}(k_s', c_s, o_s') = 1$$

where $(s_i')_{i\in[n]} = \boldsymbol{S}'$. Suppose $\mathcal{V}$ accepts π_{e_s} but $\hat{\boldsymbol{S}}$ does not come from the encryption of $\boldsymbol{S}$. The case must be that $\mathcal{P}^*$ constructs $(\hat{s}_i')_{i\in[n]} \neq (\hat{s}_i)_{i\in[n]}$ s.t. $\hat{s}_i' = \mathsf{Enc}(k_s', s_i')$ holds for $i \in [n]$. However, this breaks the semantic security of the MiMC cipher. From another perspective, suppose $\mathcal{V}$ accepts π_{e_s} but the key used to encrypt $\boldsymbol{S}$ is not k_s'. Then the case must be that $\mathcal{P}^*$ constructs a $k_s \neq k_s'$ such that $\mathsf{Open}(k_s, c_s, o_s') = 1$. This evidently contradicts the binding property of the commitment scheme Γ. The case of π_{e_s} is thus proved, which of π_{e_d} is identical without further ado.

2) Proof of privacy: We still consider the case of π_t first. Due to the zero-knowledge property of Π, any potential malicious verifier $\mathcal{V}^*$ cannot learn anything about $\boldsymbol{S}$ and $\boldsymbol{D}$ other than f and what can be directly inferred from it.

For the case of π_{e_s}, if prover $\mathcal{P}$ interacts with a malicious prover $\mathcal{V}^*$, a simulator $\mathsf{Sim}_{\mathcal{V}^*}$ can be constructed which is indistinguishable from the honest $\mathcal{P}$ from the perspective of $\mathcal{V}^*$ on input $(\hat{\boldsymbol{S}}, c_s)$. The flow of $\mathsf{Sim}_{\mathcal{V}^*}$ is as follows:

1) $\mathsf{Sim}_{\mathcal{V}^*}$ runs the simulator Sim of Π and gets $\pi_{e_s}' \leftarrow \mathsf{Sim}(\mathsf{ek}, (\hat{\boldsymbol{S}}, c_s))$. It sends the tuple $(\hat{\boldsymbol{S}}, c_s, \pi_{e_s}')$ to $\mathcal{V}^*$.
2) $\mathsf{Sim}_{\mathcal{V}^*}$ aborts.

Given the zero-knowledge property of Π, π_{e_s}' sent by $\mathsf{Sim}_{\mathcal{V}^*}$ is indistinguishable from π_{e_s} sent by an honest $\mathcal{P}$. Also, given MiMC's security against ciphertext-only attack (COA), it is computationally infeasible for $\mathcal{V}^*$ to infer k_s or $\boldsymbol{S}$ from $\hat{\boldsymbol{S}}$. From another perspective, given the hidden property of Γ, it is infeasible for $\mathcal{V}^*$ to derive k_s from c_s and decrypt $\hat{\boldsymbol{S}}$. The case of π_{e_s} is thus proved, which of π_{e_d} is identical without further ado. □

B. Security of exchange protocol

Regarding the key-secure two-phase data exchange protocol, we have the following theorem.

Theorem 5.2: Given that the commitment scheme Γ satisfies definition 2.2 and 2.3, and the NIZK scheme Π satisfies definition 2.5, 2.6 and 2.7, the key-secure data exchange protocol in IV-F has the following security properties:

- *Buyer fairness*: for any potential malicious seller $\mathcal{S}^*$, the buyer $\mathcal{B}$ must be able to learn some $\boldsymbol{D}$ such that $\phi(\boldsymbol{D}) = 1$ if its balance increases with non-negligible probability.
- *Seller fairness*: for any potential malicious buyer $\mathcal{B}^*$, if the balance of the seller $\mathcal{S}$ with whom she interacts does not increase, $\mathcal{B}^*$ must not be able to learn anything about $\boldsymbol{D}$ other than what can be directly inferred from the predicate ϕ, except with negligible probability.

1) Proof of buyer fairness: If the buyer $\mathcal{B}$ interacts with a malicious seller $\mathcal{S}^*$ and the balance of $\mathcal{S}^*$ increases, both of the conditions below must be satisfied:

1) π_p received by $\mathcal{B}$ in the *data validation phase* satisfies:

$$\mathsf{Verify}(\mathsf{vk}, (\hat{\boldsymbol{D}}, c_d, \phi), \pi_p) = 1$$

2) π_k received by $\mathcal{J}$ in the *key negotiation phase* satisfies:

$$\mathsf{Verify}(\mathsf{vk}, (k_c, c, h_v), \pi_k) = 1$$

Given the knowledge soundness of Π, if condition 1) is satisfied, except for negligible probability $\mathsf{negl}(\lambda)$, there must exist an extractor Ext_1 outputting $(\boldsymbol{D}', k', o'_d)$ s.t.

$$\phi(\boldsymbol{D}') = 1 \wedge \hat{\boldsymbol{D}} = \mathsf{Enc}(k', \boldsymbol{D}') \wedge \mathsf{Open}(\boldsymbol{D}', c_d, o'_d) = 1 \quad (4)$$

Similarly, if condition 2) is satisfied, there must exist an extractor Ext_2 outputing (k'_v, k'', o') s.t.

$$\mathsf{Open}(k'', c, o') = 1 \wedge h_v = H(k'_v) \wedge k_c = k'' + k'_v \quad (5)$$

except for negligible probability $\mathsf{negl}(\lambda)$. Given the binding property of Γ, it must be the case that $k'' = k'$ by Equation 4 and 5 except for negligible probability, and hence $k_c = k' + k'_v$.

Suppose that at the end of the interaction, the balance of $\mathcal{S}^*$ does increase yet $\mathcal{B}$ does not learn $\boldsymbol{D}'$, then the case must be that during the *key negotiation phase*, $\mathcal{B}$ computes some $k \neq k'$. To this end, $\mathcal{S}^*$ needs to construct a π_k to convince $\mathcal{J}$ that its overt k_c satisfies $k_c = k + k'_v$, which evidently contradicts the computational knowledge soundness of Π.

2) Proof of seller fairness: If the seller $\mathcal{S}$ interacts with a malicious buyer $\mathcal{B}^*$ and the balance of $\mathcal{S}$ does not increase, we can construct a simulator $\mathsf{Sim}_{\mathcal{B}^*}$ which is indistinguishable from the honest $\mathcal{S}$ from the perspective of $\mathcal{B}^*$ on input $(\hat{\boldsymbol{D}}, c_d)$. The flow of $\mathsf{Sim}_{\mathcal{B}^*}$ is as follows:

1) In the *data validation phase*, $\mathsf{Sim}_{\mathcal{B}^*}$ runs the simulator Sim of Π and gets $\pi'_p \leftarrow \mathsf{Sim}(\mathsf{ek}, (\hat{\boldsymbol{D}}, c_d, \phi))$. It sends the tuple (c_d, π'_p) to $\mathcal{B}^*$.
2) In the *key negotiation phase*, $\mathsf{Sim}_{\mathcal{B}^*}$ runs the simulator Sim of Π and gets $\pi'_k \leftarrow \mathsf{Sim}(\mathsf{ek}, (k_c, c_d, h_v))$. It sends the tuple (k_c, π'_k) to $\mathcal{B}^*$.
3) $\mathsf{Sim}_{\mathcal{B}^*}$ aborts.

Due to the zero-knowledge property of Π, π'_p and π'_k sent by $\mathsf{Sim}_{\mathcal{B}^*}$ are indistinguishable from π_p and π_k sent by the honest $\mathcal{S}$. In the case that the balance of $\mathcal{S}$ does not increase, either $\mathcal{B}^*$ aborts at the end of step 1), or $\mathcal{S}$ sees that the transaction proposed by $\mathcal{B}^*$ contains some h_v and receives k_v from $\mathcal{B}^*$ s.t. $h_v \neq H(k_v)$, in which case $\mathcal{S}$ aborts at the beginning of the *key negotiation phase*.

On the other hand, given the hiding property of Γ, it is infeasible for $\mathcal{B}^*$ to derive k from c or decrypt $\hat{\boldsymbol{D}}$. Moreover, given the collision resistance property of cryptographic hash function $H(\cdot)$, it is also infeasible for $\mathcal{B}^*$ to generate another $k'_v \neq k_v$ which satisfies $h_v = H(k'_v)$. Therefore, she cannot spoof $\mathcal{S}$ to obtain k_c in the *key negotiation phase*. □

VI. Implementation and Evaluation

In this section, we present the implementation of ZKDET and evaluate its performance in detail.

A. Implementation

We have designed and implemented a prototype of ZKDET, which contains the following components:

- ZKDET-snark: arithmetic circuits for all NIZK proofs in zkDET, along with the core logic of circuit compilation, universal setup, proof generation and verification. All arithmetic circuits are written in Circom, and the proof process is implemented using Snarkjs in JavaScript, totaling about 17k lines of code.
- ZKDET-contract: a series of smart contracts instantiating the ERC-721 specification for token management, data binding, token transferring and clock auctions. All contracts are written in Solidity, totaling about 1.2k lines of code after flattening.

The Circom library is used to prepare and compile the arithmetic circuits for NIZK in ZKDET. It uses BN-128 as the underlying elliptic curve for pairing and group operations. Snarkjs is used for setup and proof process, which supports universal setup in Plonk construct. As introduced in IV-C, MiMC-CTR and Poseidon are used as encryption and commitment primitives. Specifically, we use MiMC-p/p with round $r = 91$ and non-linear permutation of degree $d = 7$ [7]; we use x^5-Poseidon-128 with $R_F = 8$ and $R_P = 60$, referring to the recommended settings [8]. The two applications presented in IV-E are also implemented to examine the availability and performance of ZKDET in practical scenarios.

B. Evaluation on NIZK proofs

We first evaluate the performance of NIZK proofs in ZKDET. Specifically, we examine the performance of *generic transformation protocol* and *key-secure exchange protocol* in terms of universal setup, proof generation and verification. All experiments in this phase are performed under Ubuntu 20.04 on a 3.50 GHz Intel i9-11900k CPU with 64 GB of RAM.

1) Circuit pre-processing and universal setup: ZKDET provides NIZK using the Plonk construction, which utilizes an updatable universal structured reference string (SRS) to eliminate dependence on trusted setups. All arithmetic circuits are initialized using a universal ceremony and do not need to be reset even if circuits changed. We use the *Perpetual Power of Tau* ceremony conducted by Zcash and Semaphore, which is a multi-party participatory to generate verifiable universal parameters for circuits with up to 260 million (2^{28}) constraints.

Figure 5 illustrates the setup time under different number of constraints. Notice that the setup time is positively related to the number of constraints, which depends only on the input size. For reference, one of the datasets used in our experiments contains about 20,000 entries with a total size of about 1 MB, which derives a circuit with about 2^{20} constraints. The result shows that ZKDET can be set up in less than 2 minutes for regular-sized datasets using consumer-grade hardware, and the resulting circuit can be reused later.

2) Proof generation: Both π_e in the *transformation protocol* and π_p in the *exchange protocol* contain proofs of encryption and can therefore be evaluated together. Figure 6

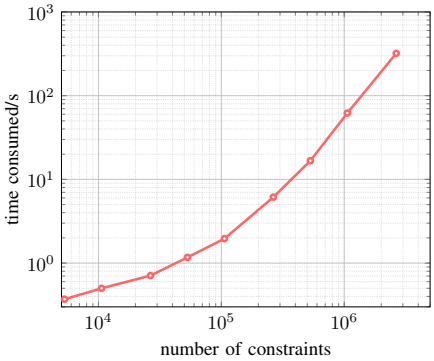

Fig. 5. Time consumed for circuit setup.

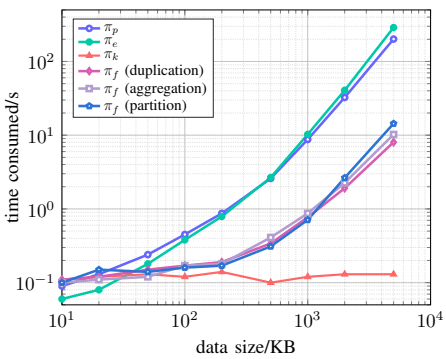

Fig. 6. Time consumed for proof generation.

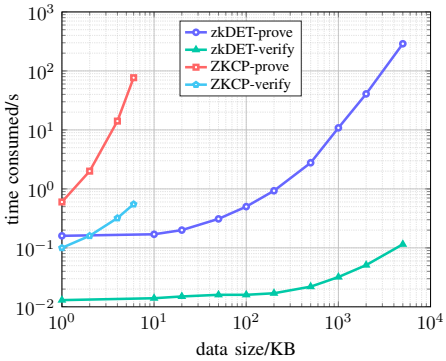

Fig. 7. Running Time of zkDET and ZKCP.

TABLE I
PROOF OF TRANSACTION FOR DATA PROCESSING APPLICATIONS.

Task	Entries / Parameters	Proof Generation Time	Proof Size
Logistic Regression	495	3.11s	2.42 KB
	1,963	21.73s	2.41 KB
	10,210	131.44s	2.45 KB
Transformer	201,163	1min29s	2.43 KB
	1,016,783	8min12s	2.41 KB

TABLE II
GAS CONSUMPTION OF SMART CONTRACTS IN zkDET.

Operation		Gas Consumed
zkDET ***Contract Deployment***		1,020,954
Verifier Contract Deployment		1,644,969
Token Minting		106,048
Token Transferring		36,574
Token Burning		50,084
Data Transformation	***Aggregation***	96,780
	Partition	83,124
	Duplication	94,012

shows the time consumed to generate these proofs, showing that for a dataset of about 5MB, it takes about 3 minutes for proof generation. Meanwhile, the π_k in the *exchange protocol* is entirely independent of the data size. As shown in Figure 6, it takes only about 120ms to generate π_k. The time consumed for proof of transformation π_t depends on the complexity of transformation f, while proofs for aggregation, partition and duplication are essentially data comparisons. Results show that proofs for these operations take only about 10s for a 5MB dataset.

We also evaluate the proof of transformation for data processing applications. Table I shows that it takes about 20s to generate a proof for logistic regression with a dataset of about 2,000 entries, and about 2 minutes when the number grows to 10,000. For proof of transformer on NLP datasets, the results show that for a model with about 1 million parameters, the time consumed is about 8 minutes.

3) Proof length and verification workload: All proofs constructed by Plonk contain only 9 elements in $\mathbb{G}_1$ and 6 in $\mathbb{F}_p$, independent of the complexity of relations to be proved [6]. Verification in zkDET is also succinct compared with that in ZKCP. It requires only 2 pairings and 18 exponential calculations on $\mathbb{G}_1$, while the latter contains 3 pairings and ℓ exponential operations in $\mathbb{G}_1$, where ℓ is the number of public inputs [6]. As shown in Figure 7, time consumed for verification remains succinct as the input size varies, remaining less than 0.1s even with large inputs.

C. Evaluation on Smart Contracts

1) Contract deployment and invocation: All smart contracts in zkDET are deployed on the Rinkeby testnet of Ethereum, including the backbone NFT contract and auction contracts. The deployment is a one-time effort, costing approximately 1,020,000 gases. Gas spent for method invocation is also evaluated, including those for token minting and transforming, as shown in Table II. Since NFTs only record and manage metadata of datasets, gas spent for invocation is quite economical.

2) On-chain proof verification: Due to their succinctness, all proof verification can be delegated to smart contracts, which reduces the workload to $\mathcal{O}(1)$ level by hardcoding group and field elements in them. Experiments show that deploying such a verifier contract costs about 1,640,000 gases and supports unlimited free verifications thereafter.

VII. RELATED WORK

A. Blockchain-based data exchange and sharing

While traditional data exchange and sharing schemes can lead to data leakage or performance bottlenecks due to centralized structures, the introduction of blockchain has been proven to solve these problems. Many studies use blockchain for data exchange in healthcare, smart vehicles, IoT and e-finance, using transactions as on-chain credentials for data assets [14], [15].

In addition to recording exchanges and committing datasets, blockchain can also provide functionalities such as transaction auditing, access control and identification for marketplaces, while providing services like dispute arbitration and data warranties for upper-layer applications [16]. Many systems have been proposed in recent years to use blockchain or smart contracts to enhance data interoperability and unlock the economic benefits of data assets [17].

Nevertheless, given the unique properties that data can be replicated and redistributed, these studies fail to address data privacy and copyright issues. While many of them propose encryption of managed datasets, the need for data authentication necessitates the existence of some super-privileged individuals [15]. Apart from that, blockchains can only provide one-way bindings of data to credentials without the ability to track data using transactions, which leads to the inability to address copyright infringement.

B. Fair data exchange

It has been well discussed that fair data exchange cannot be achieved solely by buyer and seller without the help of a trusted third party. But this limitation can be bypassed using blockchain, whose consensus mechanism can act as an arbiter.

The *Zero-Knowledge Contingent Payment* (ZKCP) protocol pioneered this field, introducing NIZK to prove that a certain predicate is satisfied without disclosing the exchanged dataset [18]. However, the initial version of ZKCP could only support a small number of simple predicates with time-consuming interactions. Several subsequent improvements significantly enhance its scalability and performance, introducing Groth16 construction for more generic predicates [10]. Nevertheless, the trusted setup of Groth16 limits its application in trustless scenarios. The recently proposed ZKCPlus scheme further improves the throughput and rate of exchanges, making it practical on consumer-grade hardware [19].

Another research direction is to circumvent expensive zero-knowledge by lightweight authenticated data structures (ADS) to provide proofs for predicates. The most representative scheme is FairSwap which constructs proofs of misbehavior based on Merkle proofs [20]. This construction allows a spoofed buyer to initiate a complaint to the arbiter contract. FairSwap performs well in optimistic scenarios where both buyer and seller remain honest. However, in the event of a dispute, the transaction cost for proof verification increases with data size, which severely limits its usefulness.

VIII. Conclusion

In this paper, we solve the problem of privacy-preserving data exchange in a public blockchain platform. To provide dynamic data traceability, privacy and exchange fairness, we propose a traceable data exchange scheme called ZKDET. Given that existing schemes cannot provide data traceability with encrypted and transformed datasets, we introduce NFT as on-chain credentials to trace all data transformations, and design an efficient generic transformation protocol to verify all transformations with zero knowledge. In addition, we propose a key-secure exchange protocol that can guarantee exchange fairness and data privacy in the scenario of public data storage. Evaluations show that ZKDET can be applied to data exchange scenarios under variant data transformations and maintain a relatively high throughput under high data volumes.

Acknowledgement

This work was partially supported by the HK RGC GRF PolyU No. 15216220 and 15217321.

References

[1] F. Stahl, F. Schomm, and G. Vossen, "The data marketplace survey revisited," ERCIS Working Paper, Tech. Rep., 2014.

[2] N. Hynes, D. Dao, D. Yan, R. Cheng, and D. Song, "A demonstration of sterling: a privacy-preserving data marketplace," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 2086–2089, 2018.

[3] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 455–471.

[4] M. Jourenko, K. Kurazumi, M. Larangeira, and K. Tanaka, "Sok: A taxonomy for layer-2 scalability related protocols for cryptocurrencies." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 352, 2019.

[5] R. Xu, G. S. Ramachandran, Y. Chen, and B. Krishnamachari, "Blendsm-ddm: Blockchain-enabled secure microservices for decentralized data marketplaces," in *2019 IEEE International Smart Cities Conference (ISC2)*. IEEE, 2019, pp. 14–17.

[6] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 953, 2019.

[7] M. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen, "Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2016, pp. 191–219.

[8] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for zero-knowledge proof systems," in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[9] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (nft): Overview, evaluation, opportunities and challenges," *arXiv preprint arXiv:2105.07447*, 2021.

[10] M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo, "Zero-knowledge contingent payments revisited: Attacks and payments for services," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 229–243.

[11] M. Campanelli, D. Fiore, and A. Querol, "Legosnark: Modular design and composition of succinct zero-knowledge proofs," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2075–2092.

[12] D. Demmler, G. Dessouky, F. Koushanfar, A.-R. Sadeghi, T. Schneider, and S. Zeitouni, "Automated synthesis of optimized circuits for secure computation," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1504–1517.

[13] F. Zhang, D. Maram, H. Malvai, S. Goldfeder, and A. Juels, "Deco: Liberating web data using decentralized oracles for tls," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1919–1938.

[14] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *2016 IEEE 18th international conference on e-health networking, applications and services (Healthcom)*. IEEE, 2016, pp. 1–3.

[15] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4660–4670, 2018.

[16] G. Zyskind, O. Nathan *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops*. IEEE, 2015, pp. 180–184.

[17] W. J. Gordon and C. Catalini, "Blockchain technology for healthcare: facilitating the transition to patient-driven interoperability," *Computational and structural biotechnology journal*, vol. 16, pp. 224–230, 2018.

[18] W. Banasik, S. Dziembowski, and D. Malinowski, "Efficient zero-knowledge contingent payments in cryptocurrencies without scripts," in *European symposium on research in computer security*. Springer, 2016, pp. 261–280.

[19] Y. Li, C. Ye, Y. Hu, I. Morpheus, Y. Guo, C. Zhang, Y. Zhang, Z. Sun, Y. Lu, and H. Wang, "Zkcplus: Optimized fair-exchange protocol supporting practical and flexible data exchange," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 3002–3021.

[20] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 967–984.