Fair^{2}Trade: Digital Trading Platform Ensuring Exchange and Distribution Fairness

Changhao Chenli, Wenyi Tang, Hyeonbum Lee, Taeho Jung (IEEE Senior Member)

***Abstract*—Online data trading is increasingly prevalent as data are becoming valuable assets. In most common conventional data trading scenarios, three parties (seller, broker, and buyer) exist, and fairness in trading is essential. This paper discusses and solves the fairness problem in two aspects. First, we consider *exchange fairness*, which requires payments and data exchanged correctly between buyers and the broker. In existing solutions, keys of encrypted data are traded. However, these solutions failed to provide a complete and secure design for validating keys' correctness unless they used generic theoretical but expensive methods, e.g., zk-SNARK. We address this security issue by designing a new key verification mechanism. We also present a novel atomic exchange protocol based on Hashed Timelock Contracts on Ethereum, reducing gas consumption compared to the existing approach. Second, we consider *distribution fairness*, which requires correctly splitting income between the broker and sellers. Straightforward solutions are impractical, i.e., sellers participating in every transaction or traversing the blockchain. Therefore, we design a verifiable statement protocol for sellers to verify the income split efficiently. Further, analysis and experimental results indicate that extra fairness properties are securely achieved, and our protocol reduces users' on-chain participation compared to state-of-the-art protocols.**

***Index Terms*—Data trading, fairness, blockchain, secret-key-sharing, atomic exchange, verifiable statement.**

I. Introduction

In the era of big data, data are considered valuable and tradeable assets because they can benefit both academia and industry. Therefore, digital data trading is increasingly prevalent, which results in the high-speed development of the marketplaces oriented by such business [1], [2]. It is estimated that the value of the data broker market was \$257 billion in 2021 and will be about \$365 billion in the year 2029 [3]. Besides B2B data brokers such as ZoomInfo, Ampliz, and Acxiom, IT companies including Google, Amazon, Alibaba, IBM, and Bloomberg also participate in the market to improve their services/products via the traded data. Many laws and regulations such as General Data Protection Regulation (GDPR) [4] and California Consumer Privacy Act (CCPA) [5] are also proposed to regulate the market of data trading, further proving the business is developing fast and widely needed.

Three parties are usually considered in a conventional data trading system (e.g., IBM Cloud Pak for Data, Snowflake Data Marketplace, etc.), including the seller, the platform (we use the word "broker" hereafter), and the buyer. We consider a similar but not the same trading scenario in this paper: a seller assigns her/his data to the broker; the broker will send the data on behalf of the seller to the buyer(s) and receive payments; after some time, the broker will split the total income during this period with the seller. Fairness is a fundamental property for every exchange or trading process. Unlike traditional data trading platforms where the broker in the middle is always trusted (i.e., large platforms are usually fully trusted based on their reputation), in our scenario, participants do not fully trust each other during the whole online trading process. Therefore, we need to ensure two fairness properties in this paper, namely exchange fairness and distribution fairness.

We first consider the exchange fairness between the broker and the buyer. A data exchange process is considered fair *iff.*: 1) the broker gets paid, and the buyer gets the desired data, or 2) neither the broker reveals the data to the buyer nor the buyer pays anything. Previous work [6] has shown that a trusted third party (TTP) who plays a role in exchanging the payment and the asset is necessary for any fair exchange protocol. There are several works [7]–[9] that leverage the smart contracts executed on the blockchain, which can replace the single TTP in traditional solutions [10], [11] between the two participants.

However, there are still non-trivial problems to be solved in the fair exchange protocols that are based on blockchain: 1) the correctness verification of the data. There might be a dispute between the two parties where the buyer claims the data are incorrect, and the broker claims the opposite. Existing works [7], [12] let the broker split the original data into multiple slices, and the buyer will choose some slices to be revealed as the correctness proof. However, these solutions lack a complete implementation and rigorous analysis on the security of the correctness check using the split-and-choose approach (e.g., how many slices should the data be split into, how many data slices should be chosen for verification, etc.); 2) the atomicity of the exchange of decryption keys and payments. To ensure the keys and the payments are exchanged correctly without repudiation, current works leverage the idea of private key locked transactions [13] to realize atomic exchange on the blockchain. Though current works are effective, the efficiency of atomic exchange can be further optimized.

Next, we further consider the distribution fairness between the broker and the seller. A revenue distribution process is fair if the broker can periodically split the income with the seller correctly. Such a revenue distribution process is not a concern in traditional systems as the broker is fully trusted. Therefore, it is necessary to design a protocol to ensure distribution fairness without a fully trusted broker in our scenario. Naïvely, the seller can verify the correctness of the revenue distribution by either participating in every transaction (in this case, the seller may even transact with buyers without a broker) or

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received April 19, 2021; revised August 16, 2021.

traversing the blockchain records to collect all her/his datasets' transaction times (note that the seller is in the same blockchain network as the broker). However, such methods will result in large overheads such as extra network burden and 24/7 liveness requirements or even privacy/security issues caused by address reuse (e.g., more evidence for retrieving the identity of the address owner or private key recovery using weak signatures) from the seller's side. (Note that the latter issues apply to all the blockchain users.) Therefore, designing a verifiable revenue distribution protocol that can guarantee distribution fairness is non-trivial.

This paper is an extension of our conference paper [14]. Both papers focus on the problem of how to ensure fairness in a digital trading scenario. The problem has been partially solved in the conference version, where we designed a random selection algorithm to facilitate the verification phase and proposed a new atomic exchange protocol to reduce costs. In this extended work, we consider a more practical scenario where a broker trades on behalf of sellers who may have insufficient resources (e.g., limited access to the Internet or inability to provide stable data availability). As the broker is not a fully trusted party, besides exchange fairness (between the broker and buyers), which can be ensured by our conference work, distribution fairness (between the broker and sellers) needs to be guaranteed in the extended work. To achieve this, we propose a verifiable statement protocol and prove that it allows sellers to verify whether the statements provided by the broker are correct. Further, both the complexity analysis and the simulation results indicate that the overhead caused by following the protocol is low. Therefore, we conclude that our extended work provides a practical solution for ensuring fairness in real-world digital trading scenarios.

In this paper, we propose a data trading platform, Fair^{2}Trade, that can ensure both exchange fairness and distribution fairness among different participants. To ensure exchange fairness during the trading between the broker and the buyer, we improved existing fair exchange protocols in two aspects. We first described a detailed design of correctness checking using the idea of the split-and-choose model. More specifically, a verifiable random-selection algorithm is used to ensure the randomness of the chosen samples so that the broker cannot cheat. Then, the atomic exchange in Fair^{2}Trade is based on our novel key-secret-sharing mechanism, which is more efficient than the current work that forces users to re-use the same random numbers in the ECDSA (Elliptic Curve Digital Signature Algorithm). Furthermore, the broker will follow a verifiable statement protocol using the idea of Merkle Tree and a hash chain structure, which can let the seller efficiently validate past transactions, and distribution fairness can also be guaranteed. Our contributions are summarized as follows:

- We designed Fair^{2}Trade, a novel data trading platform that considers and ensures both exchange fairness and distribution fairness based on a real-world online trading paradigm without the broker being fully trusted.
- Our protocol can guarantee the validity of the encrypted data and the keys sent by the broker. By designing a random sampling algorithm, we reduced the communication rounds and enhanced the security during the verification phase, compared to existing approaches [7], [12], [15].
- We proposed novel private key locked transactions based on the idea of secret sharing, which is more efficient than existing approaches based on ECDSA [7], [13].
- Our protocol allows sellers to efficiently verify the revenue distribution amount without participating in the selling process or causing a heavy blockchain storage/bandwidth burden. Security analysis also indicates that it is hard for the broker to cheat without being detected.
- Our open-source implementation shows that Fair^{2}Trade consumes less gas for ensuring exchange fairness, causes overhead for ensuring distribution fairness with certain trade-offs and reduces users' on-chain participation compared to existing works. Source codes are released for reproducibility.

The rest of the paper is organized as follows: Section II introduces related works; Section III gives the models and assumptions; Section IV describes the detailed design of Fair^{2}Trade and security analysis; Section V shows the evaluation results and Section VI summarizes the paper.

II. Related Works

Traditional fair exchange protocols usually either assume a single trusted third party (TTP) [10], [11], which suffers from the single-point-of-failure vulnerability, or are implemented in a bit-wise manner [16], [17], which suffer from the inefficiency. Recently, many fair exchange protocols have been proposed that leverage blockchains due to their decentralized nature. Since fair exchange protocols cannot remove TTP [6] without any further assumption, recent protocols can replace the single TTP with smart contracts deployed on the blockchain. Two aspects are widely discussed and explored in these works: the verification of data correctness and the atomicity of the exchange between data and payments.

For the data verification process, a function ϕ is usually assumed, with which a buyer can verify whether the dataset D to be sold satisfies ($\phi(D) = 1$) her/his demands or not ($\phi(D) = 0$) [12], [18]–[21]. For instance, FairSwap [18] assumed a public database, and buyers can compare the hash value of the traded data with the database for verification purposes. Several cryptographic-based solutions [8], [9], [22]–[24], such as using zero-knowledge proof (ZKP), zk-SNARK, homomorphic encryption (HE), plaintext checkable encryption (PCE), and chunk-by-chunk signature validation are also proposed. Besides, another type of solution [13], [15] leveraged a random sampling process where the buyer can randomly request a small portion of the dataset for verification. Note that in this paper, we follow the idea of the random sampling approaches with a design that can guarantee fewer communication rounds and no successful cheating.

Existing protocols use different approaches to ensure fairness during the exchange phase. Wan et al. [22] utilized smart contracts and hash-chain micropayments to guarantee exchange fairness. Zhao et al. [15] and Li et al. [9] also proposed a fair data trading protocol using Double-Authenticating-Preventing Signatures [25] (DAPS) to exchange secret keys and the data. Delgado-Segura et al. [7] proposed a fair exchange scheme based on private key locked transactions on Bitcoin [13]. The key idea behind existing solutions is to guarantee atomicity (or realize an atomic exchange) during the exchange process [26]. The original notion of the atomic swap was derived from Hashed Timelock Contracts (HTLC) proposed in the lightning network [27], which is widely adopted in the field of cross-chain digital assets' exchange [28]–[33]. We also follow the idea of HTLC to design our protocol for the exchange phase, which causes negligible overhead and reduces gas consumption.

Besides fairness, there are many other aspects that should be considered when designing a digital trading platform, such as pricing model [34], financial guarantee [35], security [36], privacy [37], etc. Some works also introduced trusted parties to facilitate digital trading based on different scenarios. For instance, Galteland et al. [38] introduced a trusted data manager whose signature works as datasets' endorsements. Liu et al. [39] leveraged a distributed but trusted committee for verifying data availability and managing users' anonymous credentials. Xue et al.'s work [40] supported a multiple-issuer model, and data signed by issuers are considered valid. However, these are orthogonal to this paper as we focus on fairness, and the broker is added to facilitate the trading process as a third party but without being fully trusted. In fact, after including the broker, the scenario we consider in this paper is similar to the two-sided market structure [41]. The broker, who serves as the intermediary and provides transaction facilitation between sellers and buyers, is still the most commonly considered and discussed role (besides the seller and the buyer) in many recent data market designs [42]. Therefore, the revenue distribution fairness between sellers and the broker should be further considered and guaranteed in this paper.

III. Models and Definitions

A. Threat Models

In Fair^{2}Trade, we consider three types of entities: The *seller* is the entity that will assign her/his own data to the broker and get paid. The *buyer* is the entity that will receive and pay for her/his desired data. The *broker* is the entity that will sell the data to the buyer on behalf of the seller and split the sales income with the seller periodically.

We now give two types of adversary models that we considered in this paper. First, we consider the buyer's adversary model that during an exchange process, a malicious buyer will attempt to receive the data/keys without any payment. Then, we consider the broker's adversary model: a malicious broker will attempt to get paid during an exchange process without sending the correct data/keys to the buyer. During a distribution process, a covert broker [43] will try to generate a fake statement so that s/he can gain extra revenue but will stop cheating if the misbehavior can be detected. Note that, similar to other related works [44], [45], we do not consider the situation where the broker colludes with the buyer and agrees to perform the transaction offline, or intentionally leaks data off-chain. Data trading platforms usually cannot provide security guarantees for events outside the system. However, if the data leakage is within the blockchain, there are existing works that can detect such misbehaviors [46], [47]. Moreover, if the data contain the seller's sensitive information, the seller can perform some preprocessing on the data (e.g., de-identification) to avoid personal information leakage.

Note that a dishonest seller who sells wrong or unqualified data is similar to a malicious broker during the data exchange phase. Therefore, we skip the discussion on dishonest sellers as they can be detected by our protocol in Section IV-A. Besides, we leverage a public blockchain to provide an execution environment for the protocols. Nodes in the blockchain network are in charge of verifying and processing the transactions/smart contracts according to the protocols. We also skip the discussion on misbehaving nodes since their misbehaviors (i.e., trying to infer users' data trading secrets, colluding with trading participants to destroy trading fairness, attempting to fork the blockchain, etc.) can be prevented either by our designed protocols or the consensus mechanisms underlying the blockchain systems.

B. Assumptions

We assume that: 1) Data are transferred with encryption. Without encryption, everyone in the network will be able to access the content of the data, and no one will pay for the data anymore. Therefore, the exchange between two parties can be further concluded as the exchange of the decryption keys and the payment. 2) For each exchange or distribution process, we assume that at most one of the parties (the broker or the buyer) will be malicious, and they will not collude with each other. This assumption is made because a system is hard to secure if the majority of its participants are malicious [21], [48]. 3) The method of inspecting the data correctness exists. For example, in FairSwap [18], a validation function ϕ is assumed with which the receiver can verify whether the data are required.

As aforementioned, the seller will assign the data to the broker, and the broker will continue selling the data to other buyers. To avoid trading fraud, we utilize a sample-and-check verification protocol to protect the trading process. Namely, the broker will be asked to provide a small sample of the data before selling it so that the buyer can check if it is correct. The exchange only proceeds if the sample passes the verification. Otherwise, the exchange would be terminated. Note that the data verification process is public. Therefore, other users, including other potential buyers, can also verify the data if needed. However, there are still possibilities for the broker to provide a fake dataset mixed with a portion of original data and meaningless padding that passes the correctness check and hence completes the trading fraud. We call such a potential attack as *forgery attack* since the goal for the broker is to forge a valueless dataset yet still pass the correctness check and complete the exchange. Besides, to gain extra revenue, a broker may also try to generate a fake statement that can still pass the seller's verification. We call such a potential attack as *manipulation attack* since the broker can get extra revenue distribution by generating a fake statement with fewer transactions. Detailed discussion and analysis on *forgery attack* and *manipulation attack* will be provided in Section IV-D

TABLE I
SOME SYMBOLS AND THEIR EXPLANATIONS

Symbol	Description
$\mathbb{G}$, g	finite group and its gen
D,d	one dataset to be traded and one sl
H(),H	a cryptographically secure hash functi
H(D)	the digest of *D*, which is the ha
pk,sk	public and secret key pairs in asymr
k	keys in symmetric crypto
S.Enc(),S.Dec()	symmetric encryption and c
A.Enc(),A.Dec()	asymmetric encryption and
tx	a transaction in the bloc
$\#_D$	the number of transacted ti
Root	the root of a Merkle

With the above assumptions, a malicio
refuse to accept the correct data or 2) d
correct decryption keys so that s/he could lea
of the data without any payment. Since
participate in both trading processes, a mali
therefore, 1) provide the wrong decryption k
2) launch a forgery attack, or 3) initiate a m
Besides, there are some other misbehaviors
as re-selling the data via other systems/platfo
the data so that the seller cannot detect it
such misbehaviors are out of the scope of t
refer to some related works [46], [47] for i

C. Definition of Fairness and Confidentiali

Definition 1 (Exchange Fairness)**.** *We say an exchange between a broker and a buyer is fair if: 1) the broker cannot get paid unless s/he provides the correct decryption keys to the buyer; 2) the buyer cannot access the data content unless s/he has paid sufficient money to the broker.*

Note that the definition of exchange fairness of Fair^{2}Trade shares the same idea of the *strong fairness* [49] where either both participants in a trading process will receive what they want, or nothing will be exchanged if the protocol is aborted.

Definition 2 (Data Confidentiality)**.** *We say a dataset is confidential if: 1) before it is traded, only its seller and the assigned broker know its complete content; 2) during the trading process, other users (including the buyer) have no knowledge about the dataset except a small portion of it for validation purposes; 3) when the exchange succeeds, the buyer will also have access to its complete content while others do not; 4) when the exchange fails, other users (including the buyer) only have access to the small portion.*

Definition 3 (Distribution Fairness)**.** *We say a distribution between a seller and a broker is fair if: when the broker sends a statement to the seller and shares the income with her/him according to the statement, the seller should be able to verify whether the revenue s/he receives is correct.*

IV. OUR DESIGN OF FAIR^{2}TRADE

As Figure 1 shows, each data exchange process can be divided into four phases in Fair^{2}Trade: *negotiation phase*, *verification phase*, *exchange phase*, and *commitment & statement phase*. In the negotiation phase, the seller will propose the data s/he would like to sell and find a broker for sales. The

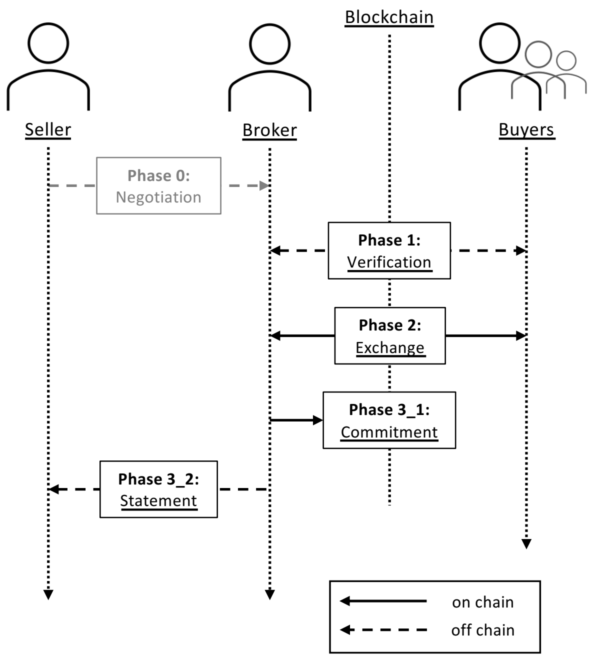


Fig. 1. Workflow of Fair^{2}Trade. In this paper, we focus on Phases 1, 2, and 3. Note that there are two parties in each phase.

broker will halt the process if s/he does not want to transact the presented data or respond to the seller to discuss the sales details. If they successfully reach an agreement, the broker will start to transact with buyers by going through the verification phase and exchange phase. After a certain time (denoted as one *epoch* hereafter), the broker will summarize the transactions during this epoch and propose a commitment to the blockchain. After the commitment is verified and recorded on the blockchain, the broker will also send a statement to each seller. Each seller can then verify her/his statement provided by the broker if needed. The seller may complain or split the income with the broker during this epoch accordingly.

Before providing the detailed design in Fair^{2}Trade, we give the definition of the Decisional Diffie-Hellman problem [50] and the algorithms in our random selection process first.

Definition 4. *Decisional Diffie-Hellman (DDH) problem in a finite group $\mathbb{G}$ with generator g is to, given the group elements (g^a, g^b, g^c), determine whether $g^c = g^{ab}$.*

There exist many finite groups where the DDH problem is known to be intractable [50] and note that our protocol can be implemented with any finite cyclic groups where the DDH problem is intractable. In this paper, we use ElGamal cryptosystem [51]. The public key of the ElGamal cryptosystem is $pk = g^{sk} \in \mathbb{G}$, where g is the generator of the group $\mathbb{G}$, and sk is the secret key randomly chosen by the seller. We omit the details of the encryption algorithms due to the space limit since they are not the most relevant to this paper.

Definition 5. *A cryptographic hash function (CHF) is a function $h : X \to Y$ where $X = \{x \in \{0,1\}^k : k \in \mathbb{N}\}$ is a set of all bit sequences of arbitrary length and $Y = \{0,1\}^l$ is the set of sequences with a specific (generally short) length of l. Inputs of h are called messages (denoted as m hereafter), and*

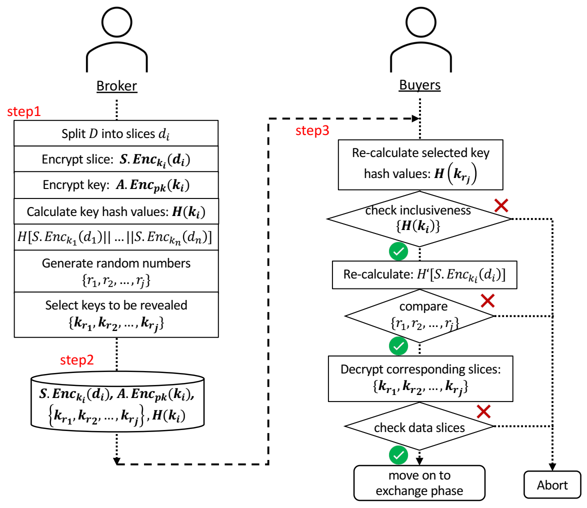

Broker
Buyers
step1
Split D into slices d_i
Encrypt slice: $\boldsymbol{S.Enc_{k_i}(d_i)}$
Encrypt key: $\boldsymbol{A.Enc_{pk}(k_i)}$
Calculate key hash values: $\boldsymbol{H(k_i)}$
$H[S.Enc_{k_1}(d_1)|| \dots ||S.Enc_{k_n}(d_n)]$
Generate random numbers $\{r_1, r_2, \dots, r_j\}$
Select keys to be revealed $\{\boldsymbol{k_{r_1}, k_{r_2}, \dots, k_{r_j}}\}$
step2
$\boldsymbol{S.Enc_{k_i}(d_i), A.Enc_{pk}(k_i),}$ $\{\boldsymbol{k_{r_1}, k_{r_2}, \dots, k_{r_j}}\}, \boldsymbol{H(k_i)}$
step3
Re-calculate selected key hash values: $\boldsymbol{H(k_{r_j})}$
check inclusiveness $\{\boldsymbol{H(k_i)}\}$
Re-calculate: $H'[S.Enc_{k_i}(d_i)]$
compare $\{r_1, r_2, \dots, r_j\}$
Decrypt corresponding slices: $\{\boldsymbol{k_{r_1}, k_{r_2}, \dots, k_{r_j}}\}$
check data slices
move on to exchange phase
Abort

Fig. 2. Workflow of verification phase: data correctness check. The broker will first split the dataset into n slices, use different symmetric keys to encrypt each slice and encrypt the symmetric keys with her/his public key pk. After the dataset encryption is finished, the broker will use the hash value of all the encrypted slices as the seed to randomly generate some numbers (e.g., using a random number generator). The broker will then choose the encrypted slices in accordance with the set of random numbers and reveal the decryption keys for these slices to the buyer. The buyer can verify whether the provided keys are included in the hashed keys' set, the correctness of the index number of the provided keys, and the content of the corresponding slices.

outputs are called digests (denoted as d *hereafter). A secure CHF should satisfy the following properties:*

1) ***Collision Resistance:*** *it is computationally infeasible to find any two distinct messages* $m \neq m' \in X$*, such that* $h(m) = h(m')$*;*
2) ***Preimage Resistance:*** *given any* $\mathsf{d} \in Y$*, it is computationally infeasible to find any* $m \in X$ *such that* $h(m) = \mathsf{d}$*;*
3) ***Second Preimage Resistance:*** *for any given* $m \in X$*, it is computationally infeasible to find* $m' \in X$ *such that* $h(m) = h(m')$ *and* $m \neq m'$*.*

Definition 6. *Our random selection process consists of two polynomial time algorithms:* $\mathsf{SeedGen}$ *and* Sample*:*

$\mathsf{SeedGen}(S.Enc(D))$ *is a deterministic algorithm that takes input as the cipher of the data* $\{S.Enc_{k_i}(d_i)_{i=1,\dots,n}\}$ *and outputs a seed* s *for the user.*

$\mathsf{Sample}(\mathsf{s}, num, n)$ *is a deterministic pseudorandom algorithm that takes input as the seed* s*, an integer* num *represents the size of the sampling set, and an integer* n *represents the size of the whole set. It outputs a set* $\{r_1, r_2, \dots, r_{num}\}$ *where each unique* $r_i \in [1, n]$*.*

Note that the results of Sample need to be pseudorandom. Besides, the broker will decide on the exact random algorithm (e.g., choose n, num, etc.) and publish it to all the users/nodes in the blockchain. By making the random algorithm publicly known to all, one can ensure the reproducibility of its results, which is important for future verification of the results.

A. Phase 1 (Verification): Data Correctness Check Requested by the Buyer and Provided by the Broker

As aforementioned in Section III-B, the verification phase is based on sample-and-check verification protocol in Fair^{2}Trade. We proposed a random sampling process to prevent malicious brokers from faking a valueless dataset yet still passing the correctness check (shown in Figure 2). Detailed steps during the verification phase are as follows:

First, the broker will divide the data D into n slices. For each slice d_i, the broker will use a different key k_i to encrypt it, which can be done by a symmetric cryptosystem such as AES [52], [53] (denoted as $S.Enc_{k_i}(d_i)$ hereafter). Then, the broker will generate a public/secret key pair (pk, sk) of ElGamal cryptosystem, encrypt each k_i, and get a set of ciphertext of the symmetric keys $\{A.Enc_{pk}(k_i)\}_{i=1,\cdots,n}$. The broker will then select some keys from the set of symmetric keys $\{k_i\}$ with the following steps:

$\mathsf{SeedGen}$: the broker will use all the encrypted slices $\{S.Enc_{k_i}(d_i)\}_{i=1,\cdots,n}$ as the input and generate a hash value $H(\{S.Enc_{k_i}(d_i)\}_{i=1,\cdots,n})$ as the seed s;

Sample: the broker will decide num based on her/his data slicing number n. Then the broker will use a hash function which takes input as the seed s concatenated by a counter for num times and generate $\{r_1, r_2, ..., r_{num}\}$.

After the process is finished, a set of verification keys $\{k_{r_j}\}_{j=1,\cdots,num}$ will be selected accordingly and sent to the buyer along with the sets $\{S.Enc_{k_i}(d_i)\}$ and $\{A.Enc_{pk}(k_i)\}$.

Upon receiving both encrypted sets and selected keys, the buyer will verify the correctness of the data with the following steps: 1) the buyer will re-calculate the hash values of the given set of verification keys $\{k_{r_j}\}$ and check whether they are included in the set of keys' hash $\{H(k_i)\}$; 2) the buyer will re-calculate $H'(\{S.Enc_{k_i}(d_i)\})$, re-generate the set of verification keys following the same random selection process, with the $H'(\{S.Enc_{k_i}(d_i)\})$ as the seed, and output a set of num indices indicating certain slices the buyer need to verify. The buyer will compare its random indices with the provided set $\{k_{r_j}\}$ and halt the verification process if both sets do not match; 3) otherwise, the buyer will decrypt the encrypted data slices using $\{k_{r_j}\}$ and verify whether the content is correct or not. If both the verification keys selection and the data content are correct, the broker and the buyer will continue to the atomic exchange of sk and the price p. Note that any secure hash can be used in Fair^{2}Trade, e.g., SHA-256 suggested by the NIST [54]. Besides, please also note that buyers will not decide the sampling process as different buyers may request different numbers of slices or use different random algorithms, which may cause data leakage concerns on the sellers' side and profit loss on both the broker and the sellers' sides. We further analyzed why the random sampling process can prevent malicious brokers from cheating in the data verification phase in Section IV-D, and discussed why we avoid letting buyers choose the samples in Section IV-E.

B. Phase 2 (Exchange): Atomic Exchange between the Broker and the Buyer for the Payment and the Secret Key sk

As aforementioned, if the data correctness is verified, the buyer will launch an atomic exchange with the broker. We follow the idea of private key locked transactions [13] as a basic framework to design our atomic exchange but with completely new and simpler mechanisms with better efficiency.

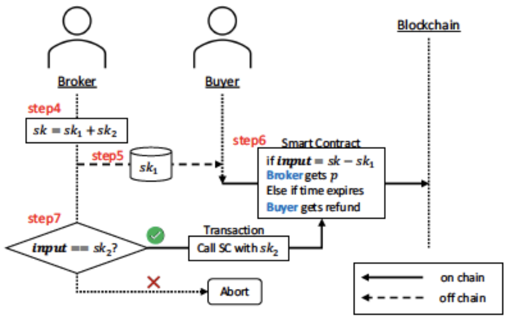


Fig. 3. Workflow of sell phase: atomic exchange for sk. The broker will split her/his secret key sk into sk_1 and sk_2 where $sk = sk_1 + sk_2$ and then send sk_1 to the buyer (privately). The buyer will create a smart contract with the condition that if the input value $input = sk - sk_1$, then an amount of p money will be paid; otherwise if no one calls the contract for some time, the buyer will get a refund. After the smart contract is deployed onto the blockchain, the broker will first check whether the condition is set correctly. If so, the broker will call the contract by providing the correct sk_2 and get paid; otherwise, the broker can just abort the process without any response.

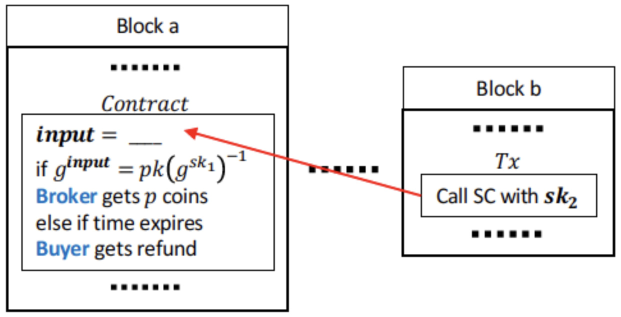


Fig. 4. An example of $Contract$ and Tx (described in Section IV-B).

In the original framework [7], Bob has an ECDSA (Elliptic Curve Digital Signature Algorithm) key pair, $\{pk_B, sk_B\}$, and Alice wants to acquire sk_B to access the data. To do this, Bob will generate a valid signature, sig_{prev}, using sk_B and send it privately to Alice. Then, Alice will create a transaction, and its output can be spent if Bob can provide another valid signature of his, which is signed using the same random value k as that of sig_{prev}. With Bob providing a valid signature as described above, Alice can recover sk_B by exploiting the ECDSA's vulnerability of reusing the same k on two different signatures. Different from the previous work, in Fair^{2}Trade, we modify the idea of secret sharing [55] to realize the atomic exchange between the broker and the buyer using a smart contract deployed on the blockchain.

To receive the pre-negotiated amount of money, the broker needs to provide sk or some information that the buyer can use to calculate sk. The broker cannot disclose sk in plaintext form because, otherwise, anyone could decrypt the data. Using a ciphertext of sk cannot solve the problem either since it will cause another key-sharing problem recursively. Therefore, we split sk into two shares, sk_1 and sk_2, and use our novel atomic exchange protocol to let the broker and buyer exchange sk with the payment. The detailed process of the exchange phase (shown in Figure 3) is as follows:

The broker randomly splits the secret sk into two shares $\{sk_1, sk_2\}$ such that $sk = sk_1 + sk_2$ (step 4). Then, the broker sends sk_1 to the buyer (step 5) via a secure channel and publishes the public key $pk = g^{sk}$. Note that the buyer knows pk and sk_1 after step 4. Since $pk = g^{sk} = g^{sk_1+sk_2} = g^{sk_1} \cdot g^{sk_2}$, it follows that $g^{sk_2} = pk \cdot (g^{sk_1})^{-1}$. Then, the buyer calculates $pk \cdot (g^{sk_1})^{-1}$ and generate the smart contract $Contract$ (step 6). After the $Contract$ is deployed onto the blockchain, the broker will verify whether the conditions are correct or not. The broker will provide sk_2 by calling the $Contract$ with a transaction Tx if the conditions are correct or abort the exchange process otherwise (step 7). The detailed content of $Contract$ and Tx are shown in Figure 4. Namely, if Tx calls $Contract$ by providing sk_2 as the input, $g^{input} = pk \cdot (g^{sk_1})^{-1}$ will be satisfied, and the $Contract$ will transfer p coins (the price for this purchase) to the broker; else if the time expires, the money will be refunded to the buyer and the broker will not get paid (which can be done via time locked transactions [7]). With both the verification phase and the exchange phase correctly and completely finished, exchange fairness can be ensured. Detailed analysis will be introduced in Section IV-D.

C. Phase 3_1 (Commitment) & 3_2 (Statement): Verifiable Revenue Distribution between the Seller and the Broker

After the exchange fairness is ensured, another protocol that can guarantee distribution fairness between the broker and her/his sellers should be designed. Though there would be no such concern if each seller could exchange with buyers directly without a broker, as we mentioned previously, it is not practical from the seller's side in real-world cases (e.g., being actively on-chain, providing stable access to the data, etc). Before we explain and provide our design reason and details, we will briefly introduce two possible solutions first:

- One naïve idea is to include both the broker and the data's corresponding seller as the recipients in every transaction. Since the broker will call the $Contract$, get verified by blockchain nodes, and trigger the transfer of the coins, the seller can receive payments without any commitment. However, unless the seller changes her/his address timely (which may also cause extra communication burden on the seller's side), a single address will be vulnerable if it repeatedly receives a lot of payments. Therefore, to guarantee security and provide convenience for sellers simultaneously, it is necessary to let the broker receive the payments from buyers first and split the income later.
- The problem now is ensuring each epoch's revenue distribution is efficient and verifiable. A following potential solution is hence to create and maintain a counter for each dataset that can record its sold times. (Note that since the seller and the broker reached an agreement during the negotiation phase, we assumed that the ratio of income split for each dataset was known to both parties. Thus, instead of the exact distribution amount, we focus on ensuring the number of transactions for each dataset (denoted as $\#_D$ hereafter) for each epoch is verifiable.) Every time the broker distributes the revenue with the sellers, s/he will fetch each seller's corresponding counter and return the results to them. However, such an approach will increase the storage burden on the

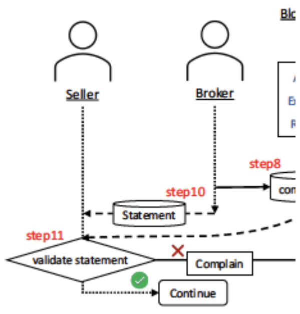


Fig. 5. Detailed process of commitment &

blockchain, where each traded
counter and a record indicating
by blockchain nodes. Extra
on the blockchain system wi
want to verify their statements,
increasing when more datasets

With the analysis of the potentia
the targets of our design are to guara
revenue distribution, provide ease c
and avoid heavy storage/bandwidth
system. Figure 5 shows the overall s
commitment & statement phase. F
will generate a commitment of all t
epoch and submit it to the block
commitment is verified and includec
the broker will generate a statemen
Sellers can verify their statements by comparing them with the on-chain commitment. Each seller can complain about the statement if it is invalid, or accept it and continue the revenue distribution process otherwise (step 11).

Before introducing the detailed protocol, we first give the notation of the linear hash chain as follows:

$$Hash_i = \begin{cases} H(D_{tx_i}), & \text{if } i = 1 \\ H(Hash_{i-1} || H(D_{tx_i})), & \text{if } i \geqslant 2 \end{cases} \tag{1}$$

When sending her/his data D to the broker, the seller will also broadcast a digest of D (here, we use the hash value $H(D)$ to represent the digest) on the blockchain. As aforementioned, we assume that both parties will negotiate the revenue distribution for each dataset. Then, the broker will transact with buyers by following the steps mentioned in Section IV-A and Section IV-B. Note that for each transaction, the buyer will also include $H(D_{tx_i})$, where D_{tx_i} represents the transacted data. With $H(D_{tx_i})$ included, blockchain nodes will be able to calculate a hash value $Hash_i$ following Equation (1) every time the broker has a new transaction successfully executed. Blockchain nodes will update and keep this hash value locally to record the accumulative transaction number per epoch and verify the commitment later (denoted as H_{curr} hereafter). On the other hand, the broker will also maintain a local list of successful transactions.

As aforementioned in Figure 5, the broker needs to generate a commitment and submit it to the blockchain for every epoch (e.g., some number of blocks). Here, we use the idea of Merkle

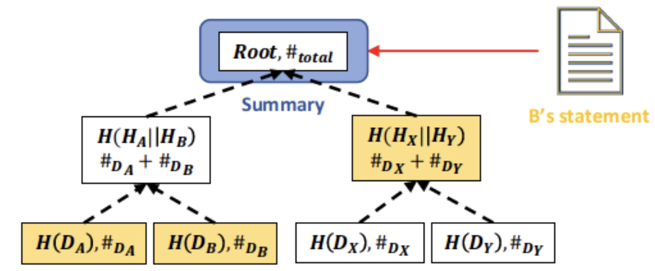


Fig. 7. An example of statement verification of seller B.

Tree as an example. As is shown in Figure 6, each Merkle Tree leaf contains the digest of a dataset $H(D_i)_{,i=1,2,\ldots,n}$ that has been transacted and the number of its sold times $\#_{D_i, i=1,2,\ldots,n}$ during this epoch. For other nodes inside the Merkle Tree, besides the hash value of its two children, the sum of sold times of the corresponding datasets will also be included. The broker will then generate a commitment that includes the Merkle leaves and the root, as well as a list that specifies the order of each digest's occurrence. When receiving the commitment, blockchain nodes will first recalculate the final result of the hash chain using the order provided in the list (denoted as H_{comm} hereafter) by following Equation (1). Note that it is necessary for the broker to clearly specify the order of $H(D_i)$ in the commitment because the result of H_{comm} is order-sensitive. The commitment will be discarded if H_{comm} does not equal H_{curr}. Otherwise, blockchain nodes will continue to verify the Merkle root and the total number $\#_{total}$. The commitment will be accepted, and only its summary (the $Root$ and the $\#_{total}$) will be included on the blockchain.

With the commitment verified and accepted, the broker will send a statement to each seller and provide a transaction id tx_{sum} that can be directed to the $(Root, \#_{total})$ stored on-chain. Thus, the statement will consist of $(tx_{sum}, \{tx_i\}, Leaf(H(D), \#_D), Proof)$, where tx_i represents the set of transactions related to D, $Leaf(H(D), \#_D)$ includes the D's digest and the number of the transactions, and the $Proof$ is the Merkle proof for the leaf (which is a list of hash values that can let the seller recalculate the Merkle root). With tx_{sum}, the seller can locate the block, compare the stored $Root$ with her/his recalculated one, and decide to continue distributing the revenue or complain based on the validation result (shown in Figure 7). Therefore, using Merkle Tree can ensure efficient statement verification without forcing sellers to traverse the blockchain records and causing limited storage complexity on the blockchain (so that extra sellers' network burden and the blockchain bandwidth can be avoided). Besides, the entry of $\#_{total}$ is included as one part of an epoch's summary so that one can quickly learn the platform's trading volume by referring to the summary transactions on-chain.

D. Security Analysis

In this section, we provide security analysis for Fair^{2}Trade. The major mechanisms in Fair^{2}Trade are based on symmetric encryption algorithms and secure hash functions. We use certain properties of these building blocks to prove security and privacy protection. Before we prove the exchange fairness and data confidentiality during the exchange phase defined in Section III-C, we first prove the robustness of the verification protocol described in Section IV-A. We call such a potential attack as *forgery* attack because the broker aims to forge a valueless dataset yet still passes the verification phase and sells it to the buyer.

Definition 7. *The verification protocol is robust if there is no polynomial time algorithm $\mathcal{A}$ for the broker to initiate a forgery attack that can pass the verification phase.*

We prove the forgery attack is impractical if the hash function used in random selection is cryptographically secure. Particularly, the hash function should be preimage-resistant and provide pseudo-randomness.

Theorem 1. *In the verification protocol, if the hash function is cryptographically secure, it is hard for any brokers with polynomial power to initiate a successful forgery attack.*

Proof. The theorem can be proven based on a polynomial reduction to the preimage-resistance of a cryptographic hash function, which indicates if such a forgery attack could be initiated efficiently (in polynomial time), we also have the ability to find a preimage of a given hash value of a cryptographically secure hash function.

First, we model the verification protocol as follows:

1) Suppose the original data D is divided into n distinct slices $\{d_{i\in[1,n]}\}$. The broker should provide a challenge set of $R_{challenge} \times n$ slices of D for the buyer to verify, where $0 < R_{challenge} < 1$ is a pre-defined challenge rate.
2) To forge a fake dataset, $D' = \{d'_{i\in[1,n]}\}$, the broker chooses a verification set V, a subset of the original data $V \subseteq D$ with size $R_{verify} \times n$, and a set of meaningless padding data Pad such that $D' = V + Pad$. R_{verify} is the rate of real data the broker would be willing to mix in the forgery for verification, where $R_{challenge} \leqslant R_{verify} < 1$.
3) The broker chooses symmetric secret keys $\{k_{i\in[1,n]}\}$ and encrypts the forged data $D' = \{d'_{i\in[1,n]}\}$ to $C' = \{c'_{i\in[1,n]}\}$, where $c'_i = S.Enc_{k_i}(d'_i)$.
4) The broker generates $\{j_1, j_2, ...j_{R_{challenge}\times n}\}$, a list of indices that are used to determine the slices to be provided for verification, according to a pseudo-random function $RndSpl(seed)$, where the $seed = H(S.Enc_{k_i}(d'_i))$ is determined by the hash value of the encrypted data, C'.
5) The victim buyer will check if all $d''_j = S.Dec_{k_j}(c'_j)$ match her/his expectation. The verification would pass *iff.* the indices list indicates that all the elements are in the verification set V.

We use the term $VerifySample(j_k, c'_{j_k})$ to represent the sample verification process in step 5), where $k \in [1, R_{challenge} \times n]$. We specify that the output of $VerifySample$ would be a set of data slices V', and a forgery attack will succeed *iff.* $V' \subseteq V$. Note that to maximize her/his profit, a malicious broker will try to minimize R_{verify}, which equals to $R_{challenge}$ and leads to $V' = V$ for a successful forgery attack (the attack is impossible if $R_{verify} < R_{challenge}$). On the other hand, if R_{verify} is large (e.g., close to 1), the attack becomes easier but also less profitable cause the broker needs to include more real data in the forged data. Therefore, we first provide our proof from the case where the broker wants to gain the maximum profit, which means $V = V'$. A quantified analysis of the difficulty of launching a successful forgery attack with different choices of $R_{verify} > R_{challenge}$ will be given later.

We now simplify the steps 2) & 3) into one polynomial algorithm $EncFD(V, Pad)$ as it only includes a single round of encryption of the forged data D'. Since in $VerifySample$, the selection of c'_{j_k} requires only finding and choosing data slices from the encrypted data C', which is determined by j_k in polynomial time, we can further simplify it into $VerifySample(j_k)$. Moreover, since the indices list is determined by a polynomial algorithm $RndSpl(seed)$, where $seed$ is further generated by $H(EncFD(V, Pad))$, therefore, the final decrypted set of data slices would be $VerifySample(H(EncFD(V, Pad))$. If there is a polynomial time algorithm to initiate a successful forgery attack, it means there is a polynomial algorithm $FindPad(V) = Pad$ which can find a valid padding dataset Pad such that $V = VerifySample(H(EncFD(V, Pad)))$. Based on this assumption, given a hash value h, we can use the forgery attack algorithm to compute $FindPad(V_h) = Pad_h$, where $V_h = VerifySample(h)$. Therefore, we will have $H(EncFD(V_h, Pad_h)) = h$ and $EncFD(VerifySample(h), Pad_h)$ will be the preimage of h. Since $EncFD$, $VerifySample$ and $FindPad$ are all polynomial algorithms, we can get the preimage of the cryptographically secure hash function in polynomial time. Therefore, the forgery attack is at least as hard as performing a successful preimage attack to a cryptographically secure hash function.

□

Even though we have proven the impossibility of an efficient forgery attack for any polynomial brokers, the attack is still possible with higher R_{verify} or even brute-forcing. The latter is because the broker will decide the details of the random sampling process (number of sampled slices, choice of random algorithm, etc.), and therefore, theoretically, the broker is able to perform a forgery attack much earlier before the verification phase. Now, we continue our quantified analysis as follows:

Suppose the cryptographically secure hash function provides nearly perfect randomness; the probability for the broker to successfully initiate a forgery attack by building a fake dataset with a fixed verification set is bounded by the following

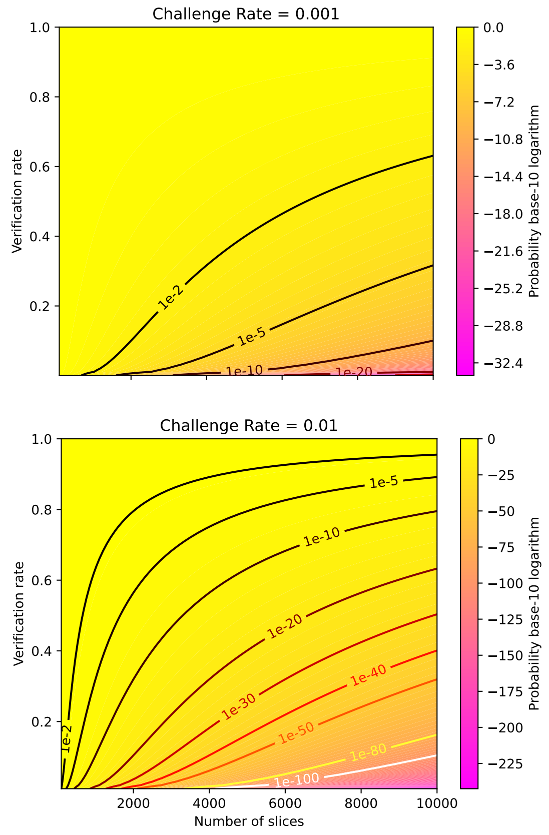


Fig. 8. Forgery attack probability under different challenge rate settings. The horizontal and vertical axes show the total slices and verification rate of the data. The colored map indicates the base-10 logarithm of the forgery attack success chance. The in-map curves are the contour lines of achieving the correlated probability on the line.

equation:

$$\Pr(\text{Forgery succeeds}) = \frac{\binom{n\times R_{verify}}{n\times R_{challenge}}}{\binom{n}{n\times R_{challenge}}}$$

in which $\binom{A}{B}$ indicates the number of B out of A combinations. As aforementioned, R_{verify} should be no less than $R_{challenge}$; otherwise, the challenge set will always be more than the verification set, and the attack will fail. This means that the broker needs to provide at least $R_{challenge} \times n$ real data slices in an attack attempt. According to the equation, the broker may choose to trade a bigger verification set size (bigger R_{verify}) for a better attack chance, but this would lead to less profit since more real data will be revealed to the buyer.

We provide the probability maps for different challenge rate settings in Figure 8. According to the figure, a larger challenge rate setting, namely more data slices asked by the buyer to verify, increases the difficulty of performing a successful forgery attack. For a challenge rate from 0.1% to 1%, the verification rate increases drastically for the same attack probability. The larger data volume also reduces the forgery attack probability. As demonstrated, the more data slices there are, the larger verification rate will be required for the broker to achieve practical attack probability. From the broker's perspective, the larger verification rate R_{verify} s/he is willing to provide, the better forgery chance $\Pr(\text{Forgery succeeds})$ s/he will get, together with more revelation of real data. For example, for a challenge rate of 1% and 6000 data slices in total, our protocol requires the broker to put at least 80% of the real data into the final dataset to achieve a $\frac{1}{10^5}$ successful forgery attack chance. For a normal polynomial power broker, the computation for brute-forcing such probability is troublesome and could also be easily mitigated by setting a time limit in the trading.

With both the robustness and the unforgeability proved, we now prove the exchange fairness we defined in Section III-C with the following theorem:

Theorem 2. *If the encryption algorithms in the verification phase and the cryptosystem in the atomic exchange phase are secure, then the proposed protocol can ensure the exchange fairness between the broker and the buyer's data exchange.*

Proof. When the broker is malicious, s/he may provide a fake sk_2 to $Contract$ and try to get the buyer's deposit. However, since the buyer has set the condition in $Contract$, the broker cannot get paid if s/he provides a wrong input. On the other hand, a malicious buyer may also generate a false condition in $Contract$. However, false conditions can be detected by the broker since s/he knows the values of sk, sk_1, and sk_2. The broker will terminate the trading by not responding to the $Contract$ if false conditions are detected, and Fair^{2}Trade can, therefore, ensure the fairness between the two parties. □

Theorem 3. *With the same assumptions in Theorem 2, if both the broker and the buyer follow the steps during the exchange process, then Fair^{2}Trade can ensure the confidentiality of the dataset during the exchange phase.*

Proof. The data traded between the broker and buyer is only transmitted in encrypted form publicly. The secret keys used in the encryption are transmitted off-chain through secure communication channels. Only the broker has the input that can satisfy the aforementioned conditions of $Contract$ since no one knows sk or sk_2 except the broker. Also, the buyer only publishes g^{sk_1} in $Contract$. Therefore, no one else can infer sk_1 due to the intractability of the discrete logarithm problem. In fact, due to the intractability of the DDH problem, g^{sk_1} is computationally indistinguishable from a random element of $\mathbb{G}$. Therefore, only the buyer can get the real data if the exchange is completed. If the exchange terminates in any step, the buyer will not get any information about the data except for the verification set s/he requests. □

Before proving the distribution fairness defined in Section III-C, we first prove the robustness of the verifiable statement protocol described in Section IV-C. We name such a potential attack as *manipulation* attack because the goal for the broker is to manipulate fake statements that can still be verified by sellers so that they will split the income based on the fake statements. As aforementioned, each statement will contain a list of transactions related to the seller's data, the data's transaction times $\#_D$ for that epoch, and a Merkle proof. Note that $\#_D$ and the data's digest $H(D)$ will jointly form the Merkle leaf. To gain extra revenue, the broker will include fewer transactions in the statements for the sellers, which will result in generating a fake Merkle leaf.

Therefore, the broker will launch the attack by either 1) generating a fake Merkle Tree with one or several fake leaves but providing a valid transaction order list that can pass the hash-chain check by the nodes or 2) submitting a correct Merkle Tree to the blockchain, generating a valid proof for each fake leaf when sending the statements to the sellers. We start our analysis from a scenario with one seller.

Definition 8. *The revenue distribution is verifiable if there is no polynomial time algorithm $\mathcal{A}$ for the broker to manipulate a statement that can provide a valid Merkle proof for a fake leaf.*

We prove it is impractical for the broker to launch a successful manipulation attack if the hash functions used in the hash chain calculation (Equation (1)) and Merkle Tree generation are cryptographically secure. Specifically, the hash functions should be second preimage-resistant.

Lemma 1. *During the commitment phase, if the hash function is cryptographically secure, it is hard for any brokers with polynomial power to manipulate a wrong Merkle Tree with a valid transaction order list that can pass the hash-chain check.*

Proof. The proof is straightforward. After each epoch, all blockchain nodes will have the same image on the value of H_{curr}. The broker knows both H_{curr} and the ordered list of transactions $\{tx\}_{ol}$ for the epoch that can be used to compute the final hash-chain result. We simplify the hash chain algorithm as a polynomial algorithm $HashChain(\cdot)$ as it just calculates the hash value by appending each element to the current result. Suppose the broker submitted a different list $\{tx\}'_{ol}$. Then, the attack will succeed if and only if $HashChain(\{tx\}'_{ol}) = H_{curr} = HashChain(\{tx\}_{ol})$. If there is a polynomial time algorithm $FindList(HashChain(\{tx\}_{ol}))$ that can provide a different list $\{tx\}'_{ol}$ whose hash-chain result $HashChain(\{tx\}'_{ol})$ is also H_{curr}, that means there would be a polynomial time algorithm that can be used to find a second-preimage collision. Therefore, it is at least as hard as finding a second preimage collision on a secure hash function for the broker to submit a fake Merkle Tree but can still pass the hash-chain check. □

Theorem 4. *With Lemma 1, we say that during the statement phase, if the hash function is cryptographically secure, it is hard for any brokers with polynomial power to manipulate a statement with a wrong leaf but a valid Merkle proof.*

Proof. The theorem can be proven based on a reduction from the second preimage-resistance of a cryptographic hash function. We model the statement generation as follows:

1) Suppose during one epoch, each dataset D_i is sold $\#_{D_i}$ times and the Merkle root $Root$ for the epoch is calculated correctly to pass the commitment phase.

2) When an epoch ends and $Root$ is recorded on-chain, the broker will generate statements for sellers. To generate a fake statement for some victim seller S_V, the broker will modify the real sold times $\#_{D_V}$ to $\#'_{D_V}$ and generate a proof p'.

3) The victim seller will reconstruct the root $Root'$ and check if it matches with the Merkle root $Root$ stored on-chain. The validation would pass *iff.* $Root = Root'$.

We use the term $MerkleVerify(Leaf, Proof)$ to represent the algorithm for validating a leaf's membership of a Merkle Tree. The fake statement will be considered valid if and only if $MerkleVerify((H(D_V), \#'_{D_V}), p') = True$. Suppose the original leaf and its correct proof is the pair $((H(D_V), \#_{D_V}), p)$ such that $MerkleVerify((H(D_V), \#_{D_V}), p) = True$. The exact process of $MerkleVerify(\cdot)$ includes recalculating the root of the given input and comparing it with $Root$ on-chain. Therefore, to generate a fake statement that can pass the check, the broker needs to find a different p' based on $\#'_{D_V}$ such that the result of root recalculation $Root'$ equals $Root$.

We simplify the Merkle root calculation algorithm as a polynomial algorithm $MklRtCal(\cdot)$ as it only includes a hash calculation for each two sibling nodes that share the same parent node. According to the process of root (re)calculation, each intermediate result (including the final result for the root value) will be calculated by the polynomial algorithm $IntmdHash(H_lc, H_rc) = H(H_l||H_r) = H_{intmd}$, where H_{intmd} represents the intermediate value for each node in the Merkle Tree and H_lc, H_rc represent the hash value of its left child and right child, respectively. The algorithm $IntmdHash(\cdot)$ is polynomial as it only runs one round hash function. If there exists a polynomial algorithm to generate a valid proof for a fake statement successfully, it means that for a given pair (H_l, H_r), there will be a polynomial time algorithm $FindChildren(Intmd(H_l, H_r)) = (H'_l, H'_r)$ which can find a different pair (H'_l, H'_r) such that $Intmd(H'_l, H'_r) = Intmd(H_l, H_r)$. (In fact, at least one such case that satisfies this equation will allow a valid proof to be created. For instance, in Figure 7, to generate a valid proof for a fake leaf of H'_B, either finding a new H'_A that makes $IntmdHash(H'_A, H'_B) = IntmdHash(H_A, H_B)$ or a new $IntmdHash(H_X, H_Y)'$ such that $IntmdHash(H(H_A||H'_B), H(H_X||H_Y)') = Root$ would be sufficient to generate a valid proof.)

Based on this assumption, given a pair of input (H_l, H_r), we can follow this algorithm to compute a different pair (H'_l, H'_r) such that $IntmdHash(H_l, H_r) = IntmdHash(H'_l, H'_r)$, and then (H'_l, H'_r) will be the second preimage of (H_l, H_r). Since $IntmdHash$, $MklRtCal$, and $MerkleVerify$ are both polynomial algorithms, we can get the second preimage of the cryptographically secure hash function in polynomial time. Therefore, with Lemma 1, launching a successful manipulation attack is at least as hard as performing a successful second preimage attack to a cryptographically secure hash function. □

Theorem 5. *If the hash functions are cryptographically secure, then the proposed protocol in the commitment & statement phase can ensure distribution fairness between the seller and the broker.*

Proof. The proof is straightforward. The provided commitment (including the Merkle leaves and the root) can be easily verified with the data digests included in each exchange. Therefore, the seller can verify the statement correctly by referring to the $Root$ stored on-chain. Since it is very hard for the broker to launch a successful statement manipulation attack, cheating on the number of transactions $\#_D$ in the statement without being detected is thus impractical within polynomial power. Therefore, a covert broker will follow the protocol correctly, $\#_D$'s correctness in the statement will be guaranteed, and the seller can easily calculate the correct revenues s/he deserved for each epoch according to the previous negotiation with the broker. □

With Theorems 2, 3 and 5, the exchange fairness, dataset confidentiality, and distribution fairness in Section III-C are fully proved.

E. Discussion

We first discuss and analyze the disadvantages that other solutions to the verification phase where the buyer will select the encrypted slices to be revealed [7], [12], [15]. In such a solution, a malicious buyer may 1) request a large number of slices to be revealed, or 2) control multiple IPs/nodes to request fewer but different slices each time. The first attack can be detected immediately when the broker is requested, and s/he can refuse to respond. The second attack cannot be discovered directly from a single request because each request looks the same from the broker's side. Fair^{2}Trade, on the other hand, can avoid such malicious buyers since the slices are chosen by the broker through running the random algorithm and we also proved in Section IV-D that it is impractical for a malicious broker to initiate a forgery attack. Moreover, without buyers participating in the sampling process, the number of communication rounds during the verification phase is reduced, which is another benefit of our protocol.

Then, we discuss a special situation where a $Contract$ is generated correctly and included in a block, but the block is forked later and no longer belongs to the blockchain. This could be a potential attack within permissionless blockchains since forks may appear at any time. A malicious buyer could, therefore, launch such an attack by following the protocols normally until the broker provides the correct sk_2. Then, s/he will attempt to fork the block that contains the $Contract$. The broker may finally get no paid but has already provided sk with the buyer. However, such attacks will happen only if the buyer can break the consensus mechanism of the blockchain, which is not realistic for normal buyers. Hence, in this paper, we assume that the blockchain consensus mechanism will prevent such attacks.

Another potential attack that may occur during the atomic exchange process is that a malicious broker will always refuse to provide the correct sk_2 to the buyer, no matter whether the $Contract$ is generated correctly or not. Even though the buyer can get refunded if such attacks happen, s/he still has extra cost compared to the broker since the buyer has already paid for the $Contract$'s deployment in the first place. Although this potential attack falls outside the scope of the broker's adversary model considered in this paper, we refer interested readers to an existing work [56], which discusses the solution to this problem. Similarly, a broker can also keep silent without sending statements to the sellers. In such cases, sellers can directly require statements from the broker or complain about the broker if there is no response from the broker.

TABLE II
TIME CONSUMPTION FOR THE BROKER AND THE BUYER IN THE VERIFICATION PHASE OF FAIR^{2}TRADE

number of data slices	AES encryption (broker's cost)	AES decryption (buyer's cost)	Sampling (cost of both)
n = 2000	551.49ms	1.3866ms	1.5172ms
n = 4000	1180.59ms	2.8791ms	2.9639ms
n = 6000	1692.95ms	4.1888ms	4.5511ms
n = 8000	2250.29ms	5.9547ms	5.9591ms
n = 10000	2918.19ms	6.7461ms	7.4291ms

Note that, besides fairness, many other aspects should also be considered when designing a digital trading platform, such as pricing models, data privacy, copyright protection, etc. Although these aspects are important, they are orthogonal to the focus of our paper, which is to ensure fairness during the trading process. Interested readers can refer to survey papers for a more comprehensive view of this field [42], [57].

V. EVALUATION WITH EXPERIMENTS

The design of Fair^{2}Trade mainly focused on three phases: the verification phase, the atomic exchange phase, and the commitment & statement phase. As is shown in Figure 1, each phase has different participants and happens either on-chain or off-chain. Therefore, we use python3 for simulating and evaluating off-chain operations (e.g., the random sampling process in the verification phase, the Merkle Tree generation/validation in the commitment & statement phase, etc.). The simulation codes are running on a computer with Ubuntu 20.04.5 LTS, Intel i7-6700 CPU (3.40GHz) with 8 GB memory. On-chain operations and functionalities (e.g., atomic exchange and commitment verification), on the other hand, are implemented with smart contracts written in Solidity on Ethereum. We use Truffle as the development environment to measure gas consumption. Our simulation codes and smart contracts are available at https://github.com/DougZaoldyeck/FairTradeExt.git.

A. Phase 1 (verification)

As aforementioned, there are many different methods that are used in different works to verify data correctness. The verification phase may happen before (e.g., buyers are provided with some samples of the data at the beginning), during (e.g., a stream of data chunks is transferred and validated sequentially [24]), and after (e.g., a Proof-of-Misbehavior (PoM) is provided to complain about the decrypted data [18], [19]) the exchange phase. In Fair^{2}Trade, we follow and improve current ideas of sampling during the verification phase because it only requires off-chain calculation/communication. Compared with other sampling-based solutions (e.g., [7], [12], [15]), a random algorithm is additionally required in Fair^{2}Trade to finish the sampling process. Therefore, the extra cost of both the broker and the buyer in the verification phase of Fair^{2}Trade comes from running the random algorithm to finish/verify the sampling process. According to the descriptions in Section IV-A, the broker needs to encrypt the sliced data and then use $H(\{S.Enc_{k_i}(d_i)\})$ as the random seed to finish sampling. The buyer will also reproduce the sampling process and decrypt the chosen slices to finish the verification phase.

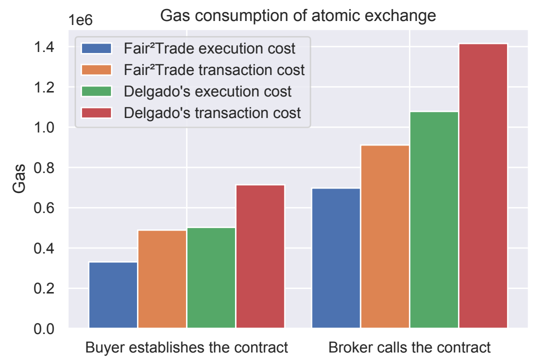


Fig. 9. Gas consumption of atomic exchange: Fair^{2}Trade and Delgado's work [7]. The horizontal axis has two values: the cost for a buyer to setup an atomic exchange and the cost for a seller to answer to the contract deployed by the buyer. For each horizontal value, we are comparing the gas consumption of the buyer/seller in Fair^{2}Trade with that in Delgado's work.

TABLE III
TIME CONSUMPTION FOR OFF-CHAIN CALCULATION IN THE COMMITMENT & STATEMENT PHASE OF FAIR^{2}TRADE

number of tx (per epoch)	Merkle Tree generation (broker's side)	statement validation (seller's side)
n = 300	0.6489μs	14.5225μs
n = 400	0.8750μs	16.3441μs
n = 500	1.0817μs	17.1053μs
n = 600	1.2309μs	22.8956μs
n = 700	1.3301μs	24.0138μs

Therefore, the overhead of Fair^{2}Trade comes from executing these operations during the random algorithm. In Table II, we include the encryption/decryption time of AES to show the total time consumption for both parties during the verification phase in Fair^{2}Trade. Here, we take $R_{challenge}$ equals to 5% as an example to present the evaluation results. Note that we omit the time of calculating $H(\{S.Enc_{k_i}(d_i)\})$ since it is negligible compared to other calculations. The simulation results show that the overhead caused by the sampling process (the process of calculating hash values as mentioned in Section IV-A) is much smaller than the AES encryption time. However, the time consumption of running the random algorithm and performing the AES decryption are close. This is because the buyer only needs to decrypt the samples during the verification phase, and the time consumption is decreased.

B. Phase 2 (Exchange)

During the exchange phase, our optimized atomic exchange protocol is implemented using smart contracts. We choose the most relevant work [7] as the baseline to compare with. As aforementioned, FairSwap [18] exchanged the secret key directly with the smart contract, and Wan et al. [22] utilized hash-chain micropayments-based smart contracts to guarantee exchange fairness. These solutions differ from ours as only part of the keys is provided, and calculation is needed for input validation. Zhao's work [15] and Li's work [9] both leveraged Double-Authentication-Preventing Signatures (DAPS) [25] to recover the secret keys. Notably, DAPS can also be instantiated efficiently on ECDSA, which means the recovery process will require a similar calculation to that of Delgado's work. Thus, we compare our work with Delgado's work only for the exchange phase. To measure the gas consumption difference between the two works, we modified from the HTLC contract [58] and Elliptic Curve arithmetic operations [59]. As is shown in Figure 9, the gas consumption of atomic exchange in Fair^{2}Trade is about 30% less than that of Delgado's work. This is because the number of operations is fewer in Fair^{2}Trade, and the complexity of computations in Fair^{2}Trade is also lower than their approach [13]. Specifically, the calculation in Delgado's design on the $Contract$ consists of Tx's signature verification and a bit-wise AND operation between two 256-bit integers to validate that the same random number is re-used. The bit-wise calculation is negligible compared to the signature verification process. In Fair^{2}Trade, the only calculation is calculating g^{sk_2} and comparing it with $pk\cdot(g^{sk_1})^{-1}$. Since the multiplication operation (major cost in Delgado's work) needs more computation power than that of the inverse operation (major cost in Fair^{2}Trade), the gas consumption is therefore reduced in Fair^{2}Trade.

C. Phase 3 (Commitment & Statement)

For the commitment & statement phase, we first analyze the time complexity for the off-chain processes, including the Merkle Tree generation (for the commitment) and the Merkle leaf validation (for the statement). As aforementioned in Section IV-C, the broker will generate a Merkle Tree to be included in the commitment and get verified by blockchain nodes. The size of the Merkle Tree is $O(n)$, where n is the total number of transactions for that epoch. The calculation complexity, on the broker's side, would be $O(n)$ for the Merkle Root generation. On the other hand, from the seller's side, to verify her/his revenue is correctly distributed, s/he just needs $O(log(n))$ time and the proof size in the statement for each seller is also limited to $O(log(n))$. We simulate the Merkle Tree generation and the leaf validation processes to evaluate users' costs during the commitment & statement phase using Python. Our results for both the broker's generation time and the seller's validation time based on different numbers of tx per epoch are shown in Table III.

Besides, we also evaluate the cost of the revenue distribution process using smart contracts. We consider two baseline works that were mentioned previously to compare with, namely, one is the naïve distribution that was mentioned in the Section IV-C where the seller will be included as the second recipient of each transaction (denoted as "Baseline without cmt" in the figure); and the other is to leverage a counter instead of the Merkle Tree and hash chain design for the commitment process (denoted as "Baseline with counters" in the figure). The first baseline approach is straightforward, where instead of the broker generating a commitment and statements per epoch, the buyer will directly pay both the broker and the seller. Namely, the buyer will include both the broker and the seller as the payment recipients when s/he deploys the smart contract (here, we can suppose all users in the platform know the revenue split ratio). Since the seller will receive the payment when the transaction between the broker and the buyer is completed, there is no need to perform an extra commitment phase. However, as we mentioned before, such a

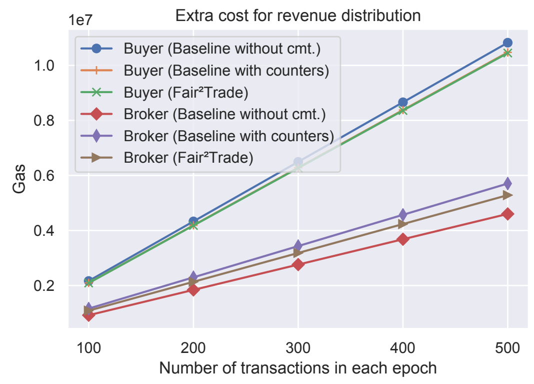


Fig. 10. Extra gas consumption of revenue distribution process in Fair^{2}Trade and two baseline approaches (one of which is a naïve distribution method where the seller is included as the second recipient in each atomic exchange and will split the payment with the broker immediately when the smart contract is called; with the other using a counter to record each sold dataset's sold times instead of the Merkle Tree and hash chain design).

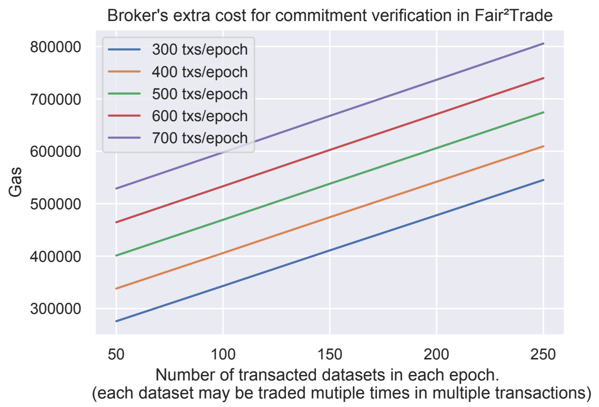


Fig. 11. Broker's extra cost for commitment verification in Fair^{2}Trade. In general, with the number of transactions fixed per epoch, the cost will grow linearly with the increasing number of traded datasets in that epoch.

method will increase the communication burden of the seller, which goes against our design goal of ease on the seller's side. The second baseline's calculation cost to increment counters for each dataset is simpler than computing a hash chain, but since each dataset has its own counter, the blockchain storage will, therefore, be increased linearly as more datasets are transacted.

In Figure 10 we can see that the buyer in the baseline work without commitment has more extra cost during the revenue distribution process compared to that of the other baseline work and Fair^{2}Trade. The cost difference among buyers is close because they need to include an extra entry in their smart contracts. The buyer in this baseline work needs to specify the seller as the second payment recipient in the smart contract. The buyer in Fair^{2}Trade and the baseline work with counters needs to include the date digest $H(D_{tx})$ in their smart contract. Therefore, the total cost on the buyers' sides in all three approaches grows linearly with the increasing number of transactions per epoch. Although for each buyer, the difference in extra cost between the approaches is small (less than 1,000 gas), sellers in the baseline approach without commitment may suffer from either the vulnerability of a repeatedly transacted address or frequent communications with the buyers to update her/his address timely. As we explained in Section IV-C, neither situation is desired in our protocol design. In Figure 11, we can see that with the total number of transactions fixed per epoch, the broker's extra cost for commitment verification grows linearly with the increasing number of transacted datasets in that epoch. This is because the more datasets are transacted in one epoch, the more leaves will the Merkle Tree have, which results in a higher cost for the Merkle Tree verification.

On the other hand, we further tested the broker's extra cost for the revenue distribution process when each dataset's transacted times changed (with the total number of transactions fixed). According to Figure 12, for an epoch with 300 transactions, the brokers' extra cost in Fair^{2}Trade and the baseline work with counters are higher than that of the baseline approach without commitment when each dataset is transacted fewer times in one epoch (less than 20 times). This is because both brokers in Fair^{2}Trade and the baseline work with counters require extra calculation for each sold dataset. Specifically, the broker in the baseline work with counters needs to set up and maintain a counter for each dataset, and similarly, the broker in Fair^{2}Trade needs to create a leaf in the Merkle tree for each sold data. Both initial operations are expensive. However, note that only one dataset is traded in each transaction, but each dataset could be traded multiple times in multiple transactions within the same epoch. Therefore, although both brokers need to consume extra gas to validate/check the commitment/counters, they only need to launch one balance transfer transaction to each seller. Thus, the more frequently each dataset is traded in one epoch, the fewer payment transactions are made by the broker.

We can also notice that for each dataset, after a certain transacted times (around 30), both brokers from Fair^{2}Trade and the baseline work with counters are lower than that of the baseline approach without commitment, but the broker in Fair^{2}Trade consumes a bit more. This is because for each transaction, the hash calculation required by Fair^{2}Trade is slightly more expensive than the counter increment. However, to ensure the statement's verifiability, each validation/check result should be recorded so that sellers can refer to it when needed. The baseline work with counters requires will need to store all datasets' sold times on the blockchain, resulting in $O(n)$ space complexity, while Fair^{2}Trade only requires $Root$ (and $\#_{total}$), which is $O(1)$. Therefore, when the seller verifies the statement, extra network resources are required on the seller's side, and the blockchain also needs to support sufficient download bandwidth. Considering the volume of the data marketplaces nowadays (e.g., one data broker holds as many as 700 billion data elements according to a report by Federal Trade Commission (FTC) [60], such extra burden is not desired by our design either.

We further compare the number of entities' on-chain participation times in Fair^{2}Trade with other works that involve three parties during a fair exchange process, including FairShare [61], a content delivery solution by He et al. [24] as well as the baseline approaches mentioned above. As shown in Table IV,

TABLE IV
COMPARISON AGAINST FAIR EXCHANGE APPROACHES WITH THREE-PARTY INVOLVEMENT IN THE SYSTEM

	Data verified	Verification method	Data access	Key access	Seller involved?	On-chain participation (n txs/epoch)	Blockchain storage overhead (n txs/epoch)
FairShare [61]	After exchange	PoM	Cloud	Fog Node	-	$O(4n)$	-
He et al. [24]	During and after exchange	Signed chunks PoM	Deliverer	Provider	-	$O((3+\zeta)\cdot n)$	-
Baseline without commitment	Before exchange	Random sampling	Broker	Broker	Yes	$O(3n)$	-
Baseline with counters	Before exchange	Random sampling	Broker	Broker	No	$O(2n+\#_{datasets})$	$O(n)$
Fair^{2}Trade	Before exchange	Random sampling	Broker	Broker	No	$O(2n+\#_{datasets})$	$O(1)$

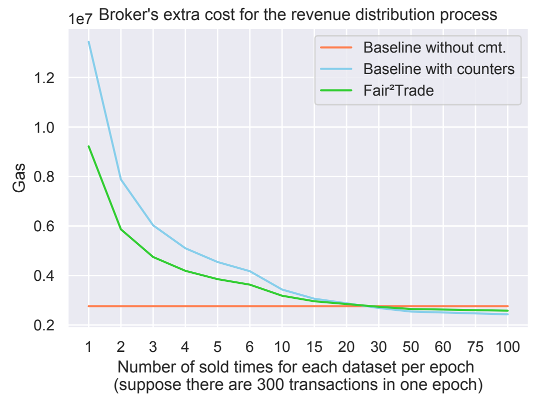


Fig. 12. Broker's extra cost for the revenue distribution process (with a total of 300 transactions per epoch). When each dataset is traded fewer times, the brokers' cost in Fair^{2}Trade and the baseline work with counters are higher because they need to submit more transfer transactions (21,000 gas/tx) as more sellers are involved in this epoch, plus the verification cost. On the other hand, the extra cost will be lower than that of the baseline approach without commitment, as fewer transfer transactions will be needed if each dataset is sold more times.

FairShare and He et al.'s work both require the client/consumer to generate a Proof-of-Misbehavior (PoM) when the decrypted data are found incorrect after the exchange. This will result in the extra $O(n)$ cost in the worst-case scenarios where each transacted dataset is complained. Besides, in He et al.'s work, they further considered *delivery fairness*, where a deliverer who indeed sends some data to the consumer would be rewarded for her/his successful data transmission. To avoid possible disputes on whether the data are delivered or not, data providers will split their data into chunks, encrypt, and sign the chunks. Each time a chunk is delivered, the consumer will validate the signature and make a response so that the deliverer can generate some proof of a successful delivery. Therefore, the total participation times of the deliverer for each transaction will be determined by the number of chunks the dataset is sliced into (denoted as ζ in the table). In the baseline works and Fair^{2}Trade, the broker will provide both the data and key access, resulting in $O(2n)$ on-chain participation times. However, in the baseline work without commitment, the seller needs to participate in each transaction (to update her/his wallet address in time to avoid using the same address repeatedly), so the total on-chain participation would be $O(3n)$. While in Fair^{2}Trade, after a commitment is verified, the broker only needs to create some transfer transactions to pay the sellers whose datasets are sold during this epoch. Therefore, the total on-chain participation of Fair^{2}Trade is $O(2n+\#_{datasets})$. Similar on-chain participation can also be seen in the baseline work with counters but will result in $O(n)$ storage overhead on blockchain in the worst case.

VI. CONCLUSION

In this paper, we present Fair^{2}Trade, a blockchain-based data trading platform that ensures both exchange fairness and distribution fairness. Specifically, Fair^{2}Trade allows buyers to verify the correctness without the risk of being deceived by the broker, accelerates the exchange process through a more efficient atomic exchange of the secret key, and enables the sellers to validate their revenue distribution with a verifiable statement protocol. The detailed verification mechanism is achieved by using random algorithms to ensure the broker cannot initiate a forgery attack within polynomial time. Our atomic exchange protocol with the key-secret-sharing mechanism is also more efficient than the ECDSA-based approaches. The verifiable statement protocol uses the hash chain and Merkle Tree structures to prevent the broker from launching any successful manipulation attack without being detected. These mechanisms can be combined together to ensure both exchange and distribution fairness in Fair^{2}Trade.

ACKNOWLEDGEMENT

This work is partially sponsored by NASA ULI under Grant No. 80NSSC23M0058, NSF under Grant No. OAC-2312973, Notre Dame International under Asia Research Collaboration Grant, Notre Dame Blockchain Working Group under Blockchain Research Grant, and IITP of the Korean Government under Grant No.RS-2022-00155220.

REFERENCES

[1] D. Chaffey and F. Ellis-Chadwick, *Digital marketing*. Pearson UK, 2019.
[2] A. Sirois *et al.*, "7 big data stocks that are bound to grow your portfolio, https://investorplace.com/2020/11/7-big-data-stocks-that-are-bound-to-grow-your-portfolio/," Nov 2020.
[3] "Data broker market: Global industry forecast (2022-2029) by data category, data type, pricing model, end use sector, and region," Aug 2022. [Online]. Available: https://www.maximizemarketresearch.com/market-report/global-data-broker-market/55670/
[4] "General data protection regulation, https://gdpr-info.eu/," Sep 2019.
[5] "California consumer privacy act (ccpa), https://oag.ca.gov/privacy/ccpa," Mar 2022.

[6] H. Pagnia and F. C. Gärtner, "On the impossibility of fair exchange without a trusted third party," Technical Report TUD-BS-1999-02, Darmstadt University of Technology . . . , Tech. Rep., 1999.

[7] S. Delgado-Segura, C. Pérez-Solà, G. Navarro-Arribas, and J. Herrera-Joancomartí, "A fair protocol for data trading based on bitcoin transactions," *FGCS*, vol. 107, pp. 832–840, 2020.

[8] Y.-N. Li, X. Feng, J. Xie, H. Feng, Z. Guan, and Q. Wu, "A decentralized and secure blockchain platform for open fair data trading," *Concurr Comput.*, vol. 32, no. 7, p. e5578, 2020.

[9] Y. Li, L. Li, Y. Zhao, N. Guizani, Y. Yu, and X. Du, "Toward decentralized fair data trading based on blockchain," *IEEE Network*, vol. 35, no. 1, pp. 304–310, 2020.

[10] F. Bao, R. H. Deng, and W. Mao, "Efficient and practical fair exchange protocols with off-line ttp," in *Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No. 98CB36186)*. IEEE, 1998, pp. 77–85.

[11] N. Asokan, V. Shoup, and M. Waidner, "Asynchronous protocols for optimistic fair exchange," in *Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No. 98CB36186)*. IEEE, 1998, pp. 86–99.

[12] Y. Chen, J. Guo, C. Li, and W. Ren, "Fade: A blockchain-based fair data exchange scheme for big data sharing," *Future Internet*, vol. 11, no. 11, p. 225, 2019.

[13] S. Delgado-Segura, C. Pérez-Solà, J. Herrera-Joancomartí, and G. Navarro-Arribas, "Bitcoin private key locked transactions," *Information Processing Letters*, vol. 140, pp. 37–41, 2018.

[14] C. Chenli, W. Tang, and T. Jung, "Fairtrade: Efficient atomic exchange-based fair exchange protocol for digital data trading," in *2021 IEEE Blockchain*. IEEE, 2021, pp. 38–46.

[15] Y. Zhao, Y. Yu, Y. Li, G. Han, and X. Du, "Machine learning based privacy-preserving fair data trading in big data market," *Information Sciences*, vol. 478, pp. 449–460, 2019.

[16] M. Blum, "How to exchange (secret) keys," *AcM Transactions on computer systems (Tocs)*, vol. 1, no. 2, pp. 175–193, 1983.

[17] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Commun. ACM*, vol. 28, no. 6, pp. 637–647, 1985.

[18] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *CCS*. ACM, 2018, pp. 967–984.

[19] L. Eckey, S. Faust, and B. Schlosser, "Optiswap: Fast optimistic fair exchange," in *Proceedings of 15th ACM ASIACCS*, 2020, pp. 543–557.

[20] R. Kumaresan, T. Moran, and I. Bentov, "How to use bitcoin to play decentralized poker," in *Proceedings of the 22nd ACM SIGSAC CCS*, 2015, pp. 195–206.

[21] M. R. Dorsala, V. Sastry *et al.*, "Fair protocols for verifiable computations using bitcoin and ethereum," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 786–793.

[22] Z. Wan, T. Zhang, W. Liu, M. Wang, and L. Zhu, "Decentralized privacy-preserving fair exchange scheme for v2g based on blockchain," *IEEE TDSC*, vol. 19, no. 4, pp. 2442–2456, 2021.

[23] R. Song, S. Gao, Y. Song, and B. Xiao, ": A traceable and privacy-preserving data exchange scheme based on non-fungible token and zero-knowledge," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 224–234.

[24] S. He, Y. Lu, Q. Tang, G. Wang, and C. Q. Wu, "Blockchain-based p2p content delivery with monetary incentivization and fairness guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 746–765, 2022.

[25] B. Poettering and D. Stebila, "Double-authentication-preventing signatures," *Int. J. Inf. Secur.*, vol. 16, no. 1, pp. 1–22, 2017.

[26] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM PODC*, 2018, pp. 245–254.

[27] R. Russell, "Lightning networks part ii: Hashed timelock contracts (htlcs)," *See https://rusty. ozlabs. org*, 2015.

[28] M. Borkowski, D. McDonald, C. Ritzer, and S. Schulte, "Towards atomic cross-chain token transfers: State of the art and open questions within tast," *DSG TU Wien (Technische Universit at Wien), Report*, 2018.

[29] M. de Vos, C. U. Ileri, and J. Pouwelse, "Xchange: A universal mechanism for asset exchange between permissioned blockchains," *World Wide Web*, pp. 1–38, 2021.

[30] M. Borkowski, C. Ritzer, D. McDonald, and S. Schulte, "Caught in chains: claim-first transactions for cross-blockchain asset transfers," *TU Wien: Technische Universität Wien, Tech. Rep*, 2018.

[31] J. A. Liu, "Atomic swaptions: cryptocurrency derivatives," *arXiv preprint arXiv:1807.08644*, 2018.

[32] Y. Guo, M. Xu, D. Yu, Y. Yu, R. Ranjan, and X. Cheng, "Cross-channel: Scalable off-chain channels supporting fair and atomic cross-chain operations," *IEEE Transactions on Computers*, 2023.

[33] S. A. Thyagarajan, G. Malavolta, and P. Moreno-Sanchez, "Universal atomic swaps: Secure exchange of coins across all blockchains," in *2022 IEEE (SP)*. IEEE, 2022, pp. 1299–1316.

[34] J. Pei, "A survey on data pricing: from economics to data science," *IEEE Transactions on knowledge and Data Engineering*, vol. 34, no. 10, pp. 4586–4608, 2020.

[35] Z. Chen, W. Ding, Y. Xu, M. Tian, and H. Zhong, "Fair auctioning and trading framework for cloud virtual machines based on blockchain," *Computer Communications*, vol. 171, pp. 89–98, 2021.

[36] J. Gao, T. Wu, and X. Li, "Secure, fair and instant data trading scheme based on bitcoin," *JISA*, vol. 53, p. 102511, 2020.

[37] W. Xiong and L. Xiong, "Data resource protection based on smart contract," *Computers & Security*, vol. 98, p. 102004, 2020.

[38] Y. J. Galteland and S. Wu, "Blockchain-based privacy-preserving fair data trading protocol," Cryptology ePrint Archive, Paper 2021/1321, 2021. [Online]. Available: https://eprint.iacr.org/2021/1321

[39] Y. Liu, X. Hao, W. Ren, R. Xiong, T. Zhu, K.-K. R. Choo, and G. Min, "A blockchain-based decentralized, fair and authenticated information sharing scheme in zero trust internet-of-things," *IEEE TC*, 2022.

[40] L. Xue, J. Ni, D. Liu, X. Lin, and X. Shen, "Blockchain-based fair and fine-grained data trading with privacy preservation," *IEEE TC*, 2023.

[41] M. Zhang, F. Beltrán, and J. Liu, "A survey of data pricing for data marketplaces," *IEEE Transactions on Big Data*, 2023.

[42] S. W. Driessen, G. Monsieur, and W.-J. Van den Heuvel, "Data market design: a systematic literature review," *Ieee access*, vol. 10, pp. 33 123–33 153, 2022.

[43] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," in *TCC, Amsterdam, The Netherlands, February 21-24, Proceedings 4*. Springer, 2007, pp. 137–156.

[44] W. Dai, C. Dai, K.-K. R. Choo, C. Cui, D. Zou, and H. Jin, "Sdte: A secure blockchain-based data trading ecosystem," *TIFS*, vol. 15, pp. 725–737, 2019.

[45] T. Jung, X.-Y. Li, W. Huang, Z. Qiao, J. Qian, L. Chen, J. Han, and J. Hou, "Accounttrade: Accountability against dishonest big data buyers and sellers," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 223–234, 2018.

[46] C. Chenli and T. Jung, "Provnet: Networked blockchain for decentralized secure provenance," in *ICBC*. Springer, 2020, pp. 76–93.

[47] C. Chenli, W. Tang, F. Gomulka, and T. Jung, "Provnet: Networked bi-directional blockchain for data sharing with verifiable provenance," *JPDC*, vol. 166, pp. 32–44, 2022.

[48] T. Rabin and M. Ben-Or, "Verifiable secret sharing and multiparty protocols with honest majority," in *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, 1989, pp. 73–85.

[49] J. Liu, W. Li, G. O. Karame, and N. Asokan, "Toward fairness of cryptocurrency payments," *IEEE Security & Privacy*, vol. 16, no. 3, pp. 81–89, 2018.

[50] D. Boneh, "The decision diffie-hellman problem," in *International Algorithmic Number Theory Symposium*. Springer, 1998, pp. 48–63.

[51] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf.*, vol. 31, no. 4, pp. 469–472, 1985.

[52] J. Daemen and V. Rijmen, *The design of Rijndael*. Springer, 2002, vol. 2.

[53] D. A. Osvik, J. W. Bos, D. Stefan, and D. Canright, "Fast software aes encryption," in *International Workshop on Fast Software Encryption*. Springer, 2010, pp. 75–93.

[54] "Nist sp 800-131a rev. 2, https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-131ar2.pdf."

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

[56] M. Lohr, B. Schlosser, J. Jürjens, and S. Staab, "Cost fairness for blockchain-based two-party exchange protocols," in *Blockchain*. IEEE, 2020, pp. 428–435.

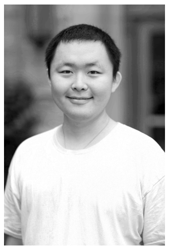
[57] F. Liang, W. Yu, D. An, Q. Yang, X. Fu, and W. Zhao, "A survey on big data market: Pricing, trading and protection," *IEEE Access*, vol. 6, pp. 15 132–15 154, 2018.

[58] Chatch, "chatch/hashed-timelock-contract-ethereum, https://github.com/chatch/hashed-timelock-contract-ethereum."

[59] Witnet, "witnet/elliptic-curve-solidity, https://github.com/witnet/elliptic-curve-solidity."

[60] J. N. . A. Ritchie and T. O. o. Technology, "Ftc agency report, https://www.ftc.gov/news-events/news/press-releases/2014/05/ftc-recommends-congress-require-data-broker-industry-be-more-transparent-give-consumers-greater?ref=hackernoon.com," May 2019.

[61] J. Sengupta, S. Ruj, and S. D. Bit, "Fairshare: Blockchain enabled fair, accountable and secure data sharing for industrial iot," *IEEE Transactions on Network and Service Management*, 2023.



Changhao Chenli Changhao Chenli is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering at the University of Notre Dame, Notre Dame, IN, USA. He received his B.S. degree and M.S. degree from Renmin University of China of information security and software engineering, Beijing, China, in 2016 and 2018 respectively. His research interest includes blockchain, data provenance, and fair exchange. His paper has won a best student paper award (IEEE BIOMETRICS COUNCIL, 2019).

Wenyi Tang Wenyi Tang is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering at the University of Notre Dame, Notre Dame, IN, USA. He received his B.S. degree and M.S. degree from Renmin University of China, Beijing, China, in 2017 and 2020 respectively. His research interest includes blockchain, data provenance, and privacy.

Hyeonbum Lee Hyeonbum Lee is currently pursuing the joint M.S/Ph.D. degree with the Department of Mathematics at Hanyang University, Seoul, Republic of Korea. He received the B.S. from Hanyang University in 2018. He was a visiting researcher with the Department of Computer Science and Engineering at the University of Notre Dame from 2022 to 2023. His research area includes cryptography and computational theory.

Taeho Jung is an assistant professor of Computer Science and Engineering at the University of Notre Dame. He received the Ph.D. from Illinois Institute of Technology in 2017 and B.E. from Tsinghua University in 2011. His research area includes data security, user privacy, and applied cryptography. His paper has won a best paper award (IEEE IPCCC 2014) and a best student paper award (BMCVAI at CVPR 2019), and two of his papers were selected as best paper candidate (ACM MobiHoc 2014) and best paper award runner up (BigCom 2015). He currently serves as associate editors of IEEE TDSC and ACM DLT.