This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2928256, IEEE Transactions on Information Forensics and Security

SDTE: A Secure Blockchain-based Data Trading Ecosystem

Weiqi Dai, Chunkai Dai, Kim-Kwang Raymond Choo, Changze Cui, Deiqing Zou, and Hai Jin

***Abstract*—Data, a key asset in our data-driven economy, has fueled the emergence of a new data trading industry. However, there are a number of limitations in conventional data trading platforms due to the existence of dishonest buyer/data broker. To mitigate these limitations, we posit the importance of a data processing-as-a-service model, which complements conventional data hosting/exchange-as-a-service model. Specifically, in this paper, we introduce a secure data trading ecosystem and present a new blockchain-based data trading ecosystem (hereafter referred to as SDTE). In the ecosystem, both data broker and buyer are not able to obtain access to the seller's raw data, as they are only getting access to the analysis findings that they require. In other words, we reduce the challenge of securing the dataset to the challenge to secure the data processing. We also build a security model to analyze the data trading market, and describe a new set of trading protocols for the entire data trading market. To demonstrate utility, we implement our proposed secure data trading platform (SDTP) on Ethereum & Intels Software Guard Extensions (SGX) and perform an in-depth analysis.**

***Index Terms*—Smart contract, Blockchain, Intel SGX, Data trading, Ethereum, EVM**

I. INTRODUCTION

THE capability to handle and process large amount of data efficiently is crucial, as such data (and their analysis) can inform decision making in both public and private sectors [1]–[4], and result in societal benefits such as increased productivity and reduced bureaucracy [5]. To facilitate the exchange of datasets, data trading platforms, also known as data exchange, have proliferated in recent years [6], [7]. Such platforms act as the bridge between sellers and buyers, and examples include Datatrading, Terbine, GXS and INFOCHIMPS.

A conventional data trading ecosystem, such as the simplified version presented in Figure 1, typically comprises three parties, namely: seller, data exchange (i.e., some middle person) and buyer. Specifically, the seller sends the dataset to a trusted data exchange platform and sets an appropriate selling price. The buyer will select a data product of interest and place an online order, similar to other e-commerce transactions. Upon receiving the buyer's payment, the data exchange platform will transmit the purchased data to the buyer, and pay the seller (after deducting the management fees or commission). However, if the raw dataset cannot be directly used by the buyer, then the buyer will need to re-process the dataset to obtain the required results satisfying the buyer's need.

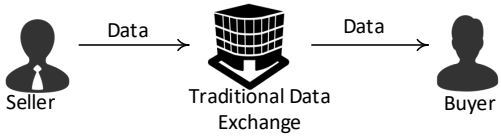

Fig. 1. Simplified conventional data trading and re-processing process

There are, however, concerns about the lack of accountability and transparency in such a centralized trading model [8]–[12], as explained below:

1) Dishonest data exchange may covertly cache and resell the seller's dataset(s), without the latter's knowledge and approval. Due to the opaque nature of such data exchange transactions, it can be challenging to detect tampering or reselling of trading information.
2) A dishonest buyer may also resell the purchased (raw) dataset, without the original seller's knowledge and approval (somewhat analogous to copyright infringement / piracy). Thus, this impacts on the profit of the original seller. In addition, a dishonest buyer may also attempt to transform the purchased (raw) dataset by performing arbitrary operations or inserting / removing data from the original dataset, in order to circumvent existing detection mechanisms.
3) The centralized data exchange model is the single point of failure / attack, where an attacker only needs to target and compromise the data exchange platform instead of multiple data owners.

A. Related Work

There have been a small number of incidents, such as those prosecuted by the U.S. Federal Trade Commission [13]. However,

This work is supported by National Natural Science Foundation of China under grant No.61602200, the Science and Technology Program of Guangzhou under grant No.201902020016, the Shenzhen Fundamental Research Program under grant No.JCYJ20170413114215614, the Guangdong Provincial Science and Technology Plan Project under grant 2017B010124001, and the Fundamental Research Funds for the Central Universities under grant No.HUST2016YXMS087. *(Corresponding author:Deqing Zou.)*

Weiqi Dai and Deiqing Zou are with National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, Big Data Security Engineering Research Center, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan, 430074, China. They also with Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen, 518057, China. (e-mail:wqdai;deqingzou@hust.edu.cn).

Chunkai Dai, Changze Cui, and Hai Jin are with National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, Big Data Security Engineering Research Center, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China (e-mail:ckdai;changzecui;hjin@hust.edu.cn).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security and the Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249-0631, USA. He also has a courtesy appointment with the School of Information Technology & Mathematical Sciences, University of South Australia, Adelaide, SA 5095, Australia (e-mail:raymond.choo@fulbrightmail.org).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2928256, IEEE Transactions on Information Forensics and Security

regulation on its own is not a viable solution as it is not realistic to police all buyers' online behavior. In addition, introducing heavy-handed regulations can stifle the market and innovation. Hence, there have been attempts in the research community to design new approaches to mitigate these limitations. Examples include the blockchain-based data trading system, the GongXinBao (GXS) system [14]. GXS facilitates direct match between buyers and sellers, where sellers can transmit data to buyers without the need to store the dataset on the platform. In addition, data sold via GXS will be uniquely tagged and the system can investigate suspected unauthorized resale activities at the request of the original sellers. However, if the dishonest buyer resells the purchased dataset offline or via a different system/platform, then clearly GXS will not be able to investigate.

Another example is AccountTrade [10], which comprises a set of accountable protocols to mitigate the risk due to dishonest buyers. Specifically, the authors described how data exchange can perform a rigorous measurement of data uniqueness (i.e., uniqueness index) and detect and punish dishonest behaviors in their approach. However, the uniqueness index is a similarity comparison mechanism (somewhat similar to iThenticate - the software used by the publishing industry to detect plagiarism), which requires data exchange to run the algorithm. Similar to the limitation in GXS, unauthorized reselling activities offline or on platforms that do not deploy the AccountTrade protocols will not be detected (similar to the limitation of journals / conferences not running iThenticate or a similar software on submitted or accepted manuscripts).

There have also been studies focusing on smart contract data feed and transaction participants' private information. For example, Town Crier (TC)'s [15] goal is to address data feeding issues in smart contracts. TC mainly uses SGX to protect the data collection process and ensure that data is not tampered with during data feed. PDFS [16] is an improvement on TC. In PDFS, the collected data comes from the blockchain, which ensures the reliability of the source data. SchellingCoin [17] allows multiple nodes to collectively submit their input data, and then selects the Nth node that submits the correct data as a special node to reward. This approach encourages nodes to provide correct data.

However, the above three studies do not protect the intellectual property of the sellers. In order to protect the personal identity information of the data seller, Zhao et al. [18] proposed a blockchain-based fair data trading protocol, which integrates ring signature, double-authentication-preventing signature and similarity learning to protect the privacy of data seller. Although the data sellers private information is protected, it cannot prevent the data buyer from reselling the received raw data.

In a different work, Shen et al. [19] proposed a remote data integrity auditing scheme that realizes data sharing with sensitive information hiding. They used a sanitizer to sanitize sensitive information and transform these sensitive information's signatures into valid ones. This allows one to verify the integrity of the sanitized file. However, the scheme does not prevent the resale of shared data.

B. Contributions and Paper Layout

In this paper, we posit the importance of analyzing and addressing the root cause rather than presenting piece-meal solutions. For example, in the existing data trading ecosystem, a buyer generally obtains the entire dataset from the seller. However, rather than having access to a complete dataset, the buyer may only need the findings from the data analysis to inform data-driven decision-making. For example, instead of having access to the seller's entire sales data for the quarter, the buyer may only need an aggregate of certain statistics for the quarter's sales data. Therefore, if the data trading ecosystem can provide such findings directly, then there is no need for the seller to transmit the entire dataset to the buyer.

Therefore, in this paper we present a new data trading ecosystem to complement the existing trading ecosystem. In other words, we move from data hosting/exchange-as-a-service to data processing-as-a-service, where the buyer is paying for the analysis of the seller's dataset. Thus, the challenge of securing the dataset is now reduced to the challenge in securing the data processing. Specifically, we build a Intels Software Guard Extensions (SGX)-based secure execution environment to protect the data processing, the source data and the analysis results. We also construct a secure data trading ecosystem (SDTE), using blockchain to prevent single-point failure. The use of blockchain also allows us to ensure that every transaction in SDTE is transparent, and facilitates the detection of any modification of the transactional information. The execution flow in SDTE is as follows (see also Figure 2):

- A buyer deploys some data processing algorithm in the form of a smart contract on our secure data trading platform (SDTP), implemented based on Ethereum. Then, the buyer identifies the data of interest, and locates prospective sellers and a number of trusted nodes to execute the data analysis contract through a broadcast contract (buyer demand broadcast contract – BDBC) on SDTP. We remark that the trusted nodes in SDTP have two Ethereum virtual machines (EVMs), where one EVM is protected by SGX and the other is a conventional EVM without SGX-protection. The traditional contract is executed on the conventional EVM of normal nodes and trusted nodes. Data analysis contract will only be executed in the SGX-protected EVM on the trusted nodes. Based on the information from BDBC, both seller and trusted node obtain the buyers needs and respond accordingly, for example via the IP address provided by the buyer. The buyer will then filter the sellers and the trusted nodes based on their information recorded in the contract information storage contract (CISC) and the speed of the complete SGX remote attestation that executes the smart contract, respectively. Then, the buyer presents the key to encrypt the data analysis result to the selected trusted nodes enclave after the successful execution of the SGX remote attestation. The buyer also sends the information relating to the selected trusted nodes information to the seller, and the latter presents the decryption key for the raw data to the selected nodes enclave. Both buyers encryption key and sellers decryption key are one-time keys, which will be sealed using the SGX seal method for latter use.
- The buyer pre-stores a sum of money on the management contract (data trading management contract, DTMC) and sets the payment amount for both seller and selected trusted

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2928256, IEEE Transactions on Information Forensics and Security

nodes upon successful execution of the data analysis contract. The seller and the selected nodes will check their reward and the contract by respectively calling DTMC and CISC. If the findings from calling DTMC / CISC align with their expectations, then the seller will send the encrypted raw data to the trusted nodes (selected by the buyer) and the selected nodes will decrypt the sellers row data and execute the data analysis contract (to analyze the raw data in the enclave).

- The buyer's selected nodes will then execute the analysis contract in SGX. The latter is a hardware-level secure execution environment, which ensures the confidentiality of data processing process, source data and data analysis results. These trusted nodes will proceed to encrypt findings from the data analysis using the buyers key in the enclave. The resultant information (hash, size and so on, but excluding the unencrypted results) will be sent to DTMC by the trusted nodes, in order to achieve a consensus on the blockchain. DTMC will count the number of different calculation results, and the outcome with the largest number of identical results will be used as the final result. After that, DTMC will reward the seller and selected trusted nodes, according to the final outcome. Finally, the trusted nodes will send the encrypted data analysis result to the buyer. The buyer retrieves the balance in DTMC and provides feedback for the seller to DTMC. DTMC will provide the buyers comment, resultant information (hash, size and so on, but excluding the unencrypted results) to CISC that can be sent to future buyers. Thus, this prevents the seller's source data from being leaked, without affecting the buyer's ability to obtain the data analysis results.

Fig. 2. Simplified secure data trading and data re-processing

In summary, our contributions are as follows:

- The design of a novel blockchain-based data trading ecosystem, SDTE, which complements existing data hosting/exchange model.
- The design of a new set of protocols for the proposed data trading ecosystem to ensure the security of data during transactions.

In the next section, we will present background materials relevant to our platform. In Section III, we present the architecture of our system and the security model of data trading market. An overview of the SDTE is presented in Section IV, prior to presenting the SDTE protocols in Section V. We also evaluate the security of the proposed SDTE in Section VI. In Section VII, we implement the secure data trading platform (SDTP) based on Ethereum, and evaluate its execution efficiency by comparing the time cost of the KNN algorithm [20] in SDTP's trusted execution environment with those of a conventional EVM environment. We conclude this paper in the last section.

II. BACKGROUND

In this section , we will briefly discuss the three basic components of SDTE, namely: smart contract, EVM and SGX.

A smart contract [21] is a computing protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow one to achieve non-repudiation in the execution of transactions without involving a third-party. In Ethereum [22], a smart contract contains a contract account, a 160-bit address, runtime bytecode and some related transactions. To deploy a contract, the developer writes and compiles the contract code into bytecode. Then, the developer uses externally owned accounts (EOAS) to build a transaction (tx) with the contract bytecode prior to broadcasting it after signing tx with his/her private key. After packing the tx into a block, all nodes in the p2p network execute using the *create* method and generate a contract account. Now, we can interact with the contract by send tx to it's address. The tx can then be packed into a block, and all Ethereum nodes will execute the block using a call method and run the contract code in the EVM. The final result will be recorded in the blockchain after consensus about this tx is achieved.

EVM is the stack-based running environment for the smart contract. During the process of running a contract, EVM reads the contract's runtime bytecode continuously and performs different operations for the different bytecodes.

SGX [23]–[25], an extension of Intel Architecture (IA), protects the execution of an application at the hardware level. For example, SGX can be used in the cloud computing environment to protect buyer data, and examples includes VC3 [26] and Haven. The core of the SGX technology is to create an area in the memory (known as EPC), where the program can create 'Enclaves' within the EPC and store key code and data into the Enclaves. Only the CPU or the application itself can access the code and data in the Enclave. SGX supports inter-platform enclave attestation to verify whether a software is running in an Enclave [24]. Attestation is a challenge-response protocol based on asymmetric cryptography. After receiving a challenge, Enclave uses an internal private key to sign the execution result of the software and reply to the challenger. After the proof is verified by the challenger, a process-provided public key can be used by the challenger to establish a secure channel with the Enclave. SGX also supports 'seal' operation, which encrypts and stores secrets outside the Enclave [24]. The sealing policies are useful for controlling the accessibility of sensitive data to future versions of Enclave.

III. ARCHITECTURE AND SECURITY MODEL

A. Overview

SDTE has three key entities, namely: buyer, seller and SDTP (blockchain nodes) – see also Figure 3. A buyer deploys one or more data analysis contracts on SDTP to analyze the sellers raw data and obtains the data analysis results. The seller is the data source, and profits from selling access of the data for analysis (whilst in the conventional model, the seller sells access to the dataset). SDTP is a secure blockchain for contract deployment, requirements matching and contract execution; and consists of trusted nodes (SGX supported) and normal nodes.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2928256, IEEE Transactions on Information Forensics and Security

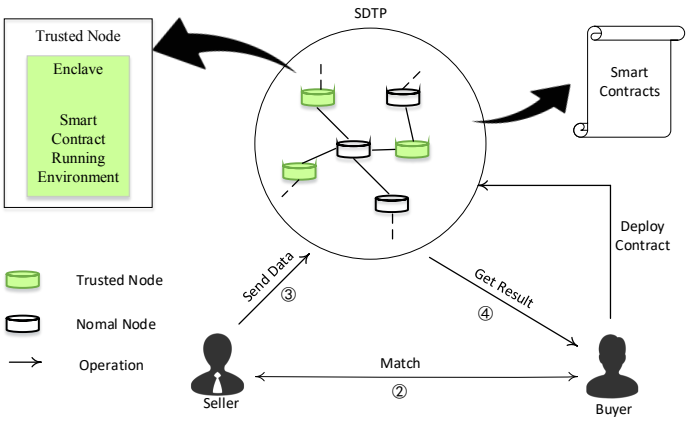

Fig. 3. An overview of SDTE

A typical workflow in SDTE is as follows. The buyer deploys his/her data analysis contracts on SDTP, finds some seller based on the data of interest, and finds SDTP's trusted nodes. Then, the seller sends the data to the trusted nodes selected by the buyer. Data analysis will be performed on the trusted nodes, and the result will be used to achieve a consensus on SDTPs contract. Only the execution results will be sent to the buyer.

B. Security Model

A summary of risks and the potential attack vectors is presented in Table I. In the table,

- **ILR** denotes "Information Leakage during Running", which occurs only during the execution of the data analysis contract on the blockchain node.
- **RF** denotes "Repudiation and Fraud", for example fraudulent activities by the buyer.
- **DF** denotes "Data Forwarding", where the seller's raw data is being forwarded to an unauthorized third-party.
- "+" indicates that the malicious actor may be the same person or there is a collusion between more than one actors in different roles. For example, in a typical transaction (i.e., seller sells the data, buyer deploys the smart contract and obtains the results, and the node is responsible for executing the smart contract and sending the result), a malicious in any of these roles can perform an attack, such as a denial of service (DoS) attack.

Examples of the risks and the potential attack vectors are as follows:

- Seller: A seller may attempt to maximize the profit by sending redundant or irrelevant data (i.e., **RF**).
- Buyer: A dishonest buyer may seek to maximize his/her profits by deliberating writing contracts that crash, stealing seller data by calling other contracts, deploying contract that cannot run in the EVM (i.e., **DF**), avoiding having to pay for using the smart contract (i.e., **RF**), and so on.
- Trusted Nodes: Trusted nodes are nodes that have SGX hardware. A trusted node in the SDTP may attempt to steal the source data and analysis result. Also, the node has super-user access to both system and physical hardware (e.g., get/delete/modify/insert arbitrary element in the stack; **ILR**). The trusted node may also cheat on the cost of running the contract or send the results to other nodes for rewards (i.e., **RF**).
- Collusion: If buyer and seller collude, They may pay less or not pay for the selected trusted node(i.e., **RF**). If seller and trusted node collude, they may want to to steal buyer's result or cheat on the cost of running contract or send their result to other nodes. So he have **RF** and **DF** attribute. The buyer and trusted node can also collude to steal the seller's data via EVM or a badly formed contract, avoid the cost of using the smart contract, and so on (i.e., **LRF**, **RF**, **DF**).
- SGX: We assume that the enclave created by SGX is sufficiently secure for executing smart contracts, and the secure channel established after the remote attestation process is reliable. We also assume that SGX sealed data is safe, in the sense that no one else other than SGX can decrypt the sealed data. It has been shown that SGX's synchronization bug can result in privacy leaks [27]. However, we do not consider SGX side-channel attacks in this work, and we refer interested reader to [28], [29] for an understanding of side-channel attacks in SGX and potential solutions. Existing solutions can be used in SDTE to prevent SGX side-channel attacks.
- DoS Attack: Attackers may invoke the contract repeatedly to exhaust network communication resources in SDTP.
- Network Communication: The entity controlling the network (e.g., Internet Service Providers, or a man-in-the-middle attacker) may tamper with or delay transmission during the session between seller, buyer and nodes.
- Ethereum: The integrity of the contracts and transactions is ensured, but not confidentiality.

TABLE I
MALICIOUS ROLES AND POTENTIAL ATTACKS

	ILR	RF	DF	DOS
Seller	-	√	-	√
Buyer	-	√	√	√
Trusted Node(TN)	√	√	-	√
Seller+Buyer	-	√	-	√
Seller+TN	√	√	-	√
Buyer+TN	√	√	√	√
Seller+Buyer+TN	-	-	-	-

In the next section, we will present our SDTE.

IV. SDTE OVERVIEW

A. Three Key Security Features

SDTE has three key security features, as follows:

Resisting data theft from nodes. In SDTP, the processes that execute the smart contract, source data and data analysis results need to be protected. Specifically, in the process of running the smart contract, the generated intermediate data should be protected. The seller's source data must not be stolen by any party during the entire transmission and execution. As for data analysis results, the ideal scenario is that the calculation result can only be obtained by buyer.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2928256, IEEE Transactions on Information Forensics and Security

Prevent repudiation and fraud. In SDTP, seller and selected nodes may be cheated by the buyer as previously discussed. At the same time, they may also deceive the buyer. Thus, we need to ensure that the buyer must pay the reward and only the honest seller and nodes will be paid, as well as the buyer cannot escape from paying the cost of executing the smart contract and malicious nodes cannot cheat the buyer to obtain the reward. Also, we must ensure the buyer's legal interest. We need to also consider the fact that both the buyer and, the seller may be malicious. For example, the seller may attempt to send redundant data for extra profit or send irrelevant data to trick the buyer. Therefore, SDTE allows the user to determine whether the seller is good or bad when selecting the seller and places a limit on the maximum profit to prevent sellers from sending additional data after selecting the seller.

Prevent malicious contract. Since Ethereum provides CALL/CALLCODE methods to call external address with arbitrary arguments, malicious smart contracts may attempt to (illegally) transmit input data to external address; thus, resulting in other contract(s) to obtain the seller's raw data without paying and without authorization. The buyer may also write a contract to output the source data to himself/herself directly. Therefore, the contracts' CALL/CALLCODE invocation of the EVM in enclave should be limited to prevent the sending of raw data to other contract(s) and some measures should be taken to prevent output source data

B. Design Principles

SDTE is designed around the three key security features, and we will explain the core design.

Resisting data theft from nodes. As previously discussed, SDTE contains three key data types, namely: the intermediate data generated during the process of running smart contract, the source data and the final data analysis results. Prior to executing the data analysis contract, raw data from the seller needs to be sent to the buyer-selected trusted node. In order to protect the raw data, the seller's data needs to be encrypted before transferring to the trusted node and then decrypt in the SGX security environment of the trusted node. In this way, the raw data cannot be leaked before being processed by the smart contract. In Ethereum, the running environment of smart contracts is EVM, which limits access from inside to outside. However, EVM does not limit access from outside to inside, so the node may steal the intermediate data, source data and analysis results during decryption or execution. Therefore, we place the decryption process of raw data and EVM in an enclave created by SGX, in order to prevent the (malicious) node from obtaining the raw data, intermediate data, analysis results.

After the analysis results is available from SGX, the trusted nodes will send the (encrypted) results to the buyer (to prevent the leakage of the results). In other words, the analysis result will not be leaked to a third-party prior to the buyer obtaining the results. Since the seller keeps the key for decrypting the raw data and the buyer keeps the key for encrypting the analysis results, a secure key transmission process is necessary for the node to get the related keys into the SGX environment. Therefore, we need to establish a secure channel through SGX remote attestation to transfer the respective keys. In this way, the seller's and buyer's keys can be directly passed into the enclave and stored outside the enclave using the SGX SEAL method. After getting the related keys, the encrypted data sent by the seller is decrypted in the enclave and the result required by the buyer is encrypted in the enclave. Thus, we rely on the secure encryption and SGX to ensure the security of the critical data.

Non-repudiation and fraud prevention. Although critical data is protected, malicious buyers may refuse to pay the seller and the nodes. For example, the malicious nodes may falsify the results and collude with other selected nodes to obtain the buyer's reward. A malicious seller may also attempt to receive additional rewards by sending additional irrelevant data to the smart contracts. In order to ensure that seller, buyer and trusted node can honestly fulfill their obligations, we introduce a data trading manager contract (DTMC) as a guardian. DTMC is a traditional Ethereum smart contract, which will honestly record incoming information.

Before executing the data analysis contract, the buyer should deploy the contract written by himself/herself on the SDTP. Then, the buyer will make an offer price for the seller and the trusted node, which is made public and written into DTMC. Also, the buyer will pre-store a certain amount of Ether to DTMC, as commitmentt. Before the seller sends data to the data analysis contract, (s)he will query DTMC and check whether the offer price is satisfied. If the buyer changes the reward to the seller, then the seller will be notified by the DTMC. In the same way, the nodes will also send a transaction to the DTMC to check whether the reward to be received has changed before executing the smart contract.

During executing the contract, private data will not be leaked because of the underpinning SGX protection. After executing the contract, the nodes will send execution result information to DTMC and a consensus will be made in DTMC to calculate the final result. The way to get the final result is that the largest number of identical results will be used as the final result. Then, DTMC will give the reward to seller and selected trusted nodes according to the final result. This ensures that only honest nodes can be rewarded. The buyer can rate the seller, which will be discussed later. To ensure that the results calculated in the enclave have not been tampered by the nodes during transmission, the enclave will generate an internal key and sign the execution result. To verify this signature, DTMC will obtain the related key from the buyer who can get the key during SGX attestation. It is worth noting that this key pair is generated within the enclave of the trusted node. After the private key is generated in the enclave, it is stored using the SGX sealing method; thus, the trusted node cannot get the private key. The public key is sent directly to the seller. If the node changes the public key, the public key stored by the seller into the DTMC will also change. Then, the DTMC verifies that the signature of the trusted node's enclave result fails. Consequently, the trusted node will not be able to claim the reward.

The seller may send redundant data to obtain extra rewards, which is clearly against the buyer's interest. Therefore, in SDTE, the buyer can set a cap for the maximum reward in DTMC, and the user can rate the seller, using existing approaches of [30], [31]. Specifically, the users can make a honest evaluation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2928256, IEEE Transactions on Information Forensics and Security

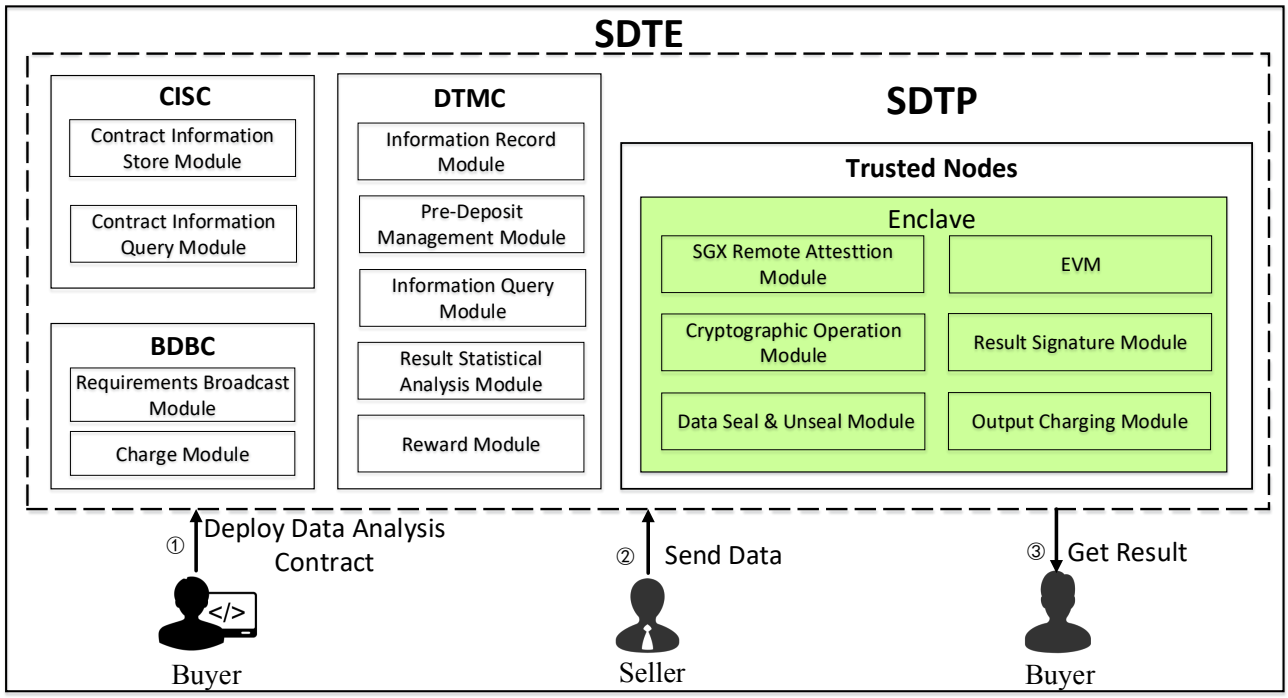

Fig. 4. Smart contract deployment in SDTP

of the seller in the dedicated smart contract (CISC) on SDTE, which allow other buyers to view the comments on the contract to determine whether they want to acquire services / data from the particular seller (similar to existing restaurant reviews on sites like Yelp). The approach in [30] divides the opinion spam into type 1 (untruthful opinion), type 2 (reviews on brands only), and type 3 (non-reviews). Both types 2 and 3 can be detected by using supervised learning with manually labeled training examples. As for type 1 comments, the authors used duplicate spam reviews as positive training examples and other reviews as negative examples to build a model. The other approach in [31] fused genre identification, psycholinguistic deception detection (LIWC2007), and text categorization, to select opinion spam and encourage buyer to make a honest comment. In the context of this paper, by combining these two existing commenting approaches and the blockchain, any malicious / dishonest behavior in the system will be revealed. With a bad reputation, it is unlikely that the particular seller will be chosen by future buyers. In addition, as all transactions in SDTP are transparent, buyers can query transactions on the blockchain and make more accurate determination in conjunction with smart contracts that store other buyer's reviews. Therefore, this mitigates the limitations in existing reviewing systems where other users can be paid to provide fictitious reviews, as well as minimizing the risk of some centralized exchange colluding with a data owner to influence the ratings.

Prevent malicious contract. Since contracts could transmit data to other contracts by invocations, the buyer may leave a backdoor in the data analysis contract. To prevent this from occurring, EVM encapsulated in the enclave forbids calls between contracts in SDTE. Moreover, the malicious buyer may want to use contract to directly output the seller's raw data. Therefore, the contract running in nodes' enclave will additionally charge the buyer based on size of the output. In other words, more output means a higher charge. If the buyer directly outputs the complete (or the relevant part of) source data, then the cost is significant. This cost is equivalent to the cost of obtaining copyright of the seller's complete (or the part of) source data. To further improve the security of source data, we are also considering using existing solutions, such as libdft [32] and CloudFence [33], to track the incoming data streams. Specifically, if the seller does not agree to output the complete (or the relevant part of) source data, then within the enclave, the decrypted source data is tracked to prevent the complete (or the relevant part of) source data from being produced as output. As there are existing solutions, data stream tracking is not the focus of this paper and we will implement data stream tracking in future work. Currently, we only implement the output charge module in SDTP.

V. SDTE PROTOCOL

To achieve the three key security features discussed in the preceding section, we will now present the module design of SDTE. The interaction process of these modules are divided into four stages, namely: contract deployment, requirements matching, execution preparation, and contract execution.

A. Module Design

SDTE contains four important function requirements, namely: contract register, requirement matching, financial management and trusted environment, which are presented as contract information storage contract (CISC), buyer demand broadcast contract (BDBC), data trading management contract (DTMC) and trusted nodes separately. Here, we will discuss the design of those function modules first – see also Figure 4.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2928256, IEEE Transactions on Information Forensics and Security

- CISC. CISC stores different seller's information through Contract Information Store Module, which is only called by DTMC to store buyer's comments on the seller after each data transaction and the execution information (hash, size and so on, but not including the unencrypted results). Existing approaches, such as those of [30], [31], can be used. The Contract Information Query Module in CISC help prospective buyers select appropriate sellers based on the contract information in CISC.
- BDBC. Requirements Broadcast Module receives and broadcast buyers' demands to Ethereum nodes, which enables seller and trusted nodes to know buyers' requirements. The Charge Module in BDBC is used to charge the buyer Ether to prevent denial of service (DoS) attacks.
- DTMC. DTMC records seller's dataset specifications and bid price (for both seller and trusted node) from the buyer by Information Record Module. Pre-deposit Management Module requires buyers to deposit some Ether to pay for seller and trusted nodes after execution. DTMC also provides a query interface named Information Query Module for trusted nodes and seller to check their reward. Result Statistical Analysis Module analyzes execution results and the most numerous identical results are the final result. Reward Module is responsible for sending Ether to related address(include seller and trusted nodes) based on the final result.
- Trusted Nodes. SGX Remote Attestation Module is used to authenticate a node's SGX environment remotely and establish a secure channel for data transmission. Enclave in a trusted node can use Data Seal & Unseal Module to securely store secret data outside or read the sealed data into Enclave. Cryptographic Operation Module is used to decrypt the input data from seller. Also, it encrypts buyer's data analysis result. EVM processes seller's data by using buyer's data analysis contract. Output Charging Module charges based on the size of EVM's output. Result Signature Module signs the result in Enclave to prevent the node from falsifying it.

Note that BDBC should not be implemented in DTMC because buyer needs to pre-deposit some Ether into DTMC when he calls it and there may be a long negotiation process before buyer and seller reach a agreement. Thus, if do so, buyer's money may be locked in DTMC for a long time.

B. Summary of Notations

A summary of notations used in our proposal is presented in Tables II and III.

C. Contract Deployment

- The deploy process of data analysis smart contract is same as traditional deploy process of Ethereum.The address obtained after deploying the data analysis contract is $A_{contract}$

D. Requirements Matching

In this section, the buyer needs to find appropriate data analysis contract and data source through SDTP. Also, buyer needs to select trusted nodes to execute the data analysis contract. We define this process as requirements matching and Figure 5 shows more detail about it.

- Step (1), The buyers' demand can be divided into three main parts: I. dataset specifications II. bid price, including P_{seller}, T_{seller}, P_{node} and T_{node} III. the number of trusted nodes to execute $C_{analysis}$. P_{seller} and P_{node} denote how much reward after processing unit data seller and node can get. T_{seller} and T_{node} is the maximum reward which seller and node can get. At the same time, buyer will also broadcast IP_{buyer} and $A_{contract}$ to prepare for the following process.
- Step (5), K_{buyer} is transmitted to the node's Enclave and it will use SGX seal method to store <ID, IP_{buyer}, K_{buyer}>, K_{buyer} is used to encrypt data analysis results. (to know more about the reason for sealing the data, section IV-A).

TABLE II
SUMMARY OF ROLES

Role	Explanation
A_{role}	Ethereum address of role
K_{role}	An AES-256 Key, role created
P_{role}	The price of unit data, as role wish
T_{role}	The maximum cost of purchasing data role can afford
IP_{role}	The IP address of role
Pri_{role}	Private key created by role to sign
Pub_{role}	Public key created by role to verify the signature generated by role

TABLE III
OTHER NOTATIONS

Notation	Explanation
ID	An identifier for a data analysis transaction,ID=Hash(K_{buyer}+IP_{buyer})
(m)n	A section of ciphertext generated by using n encrypt m
<a,b,c>	A map stores a,b,c. a is the key
CISC	Code Information Storage Contract
BDBC	buyer Demand Broadcast Contract
DTMC	Data Trading Management Contract
$C_{analysis}$	Data analysis smart contract

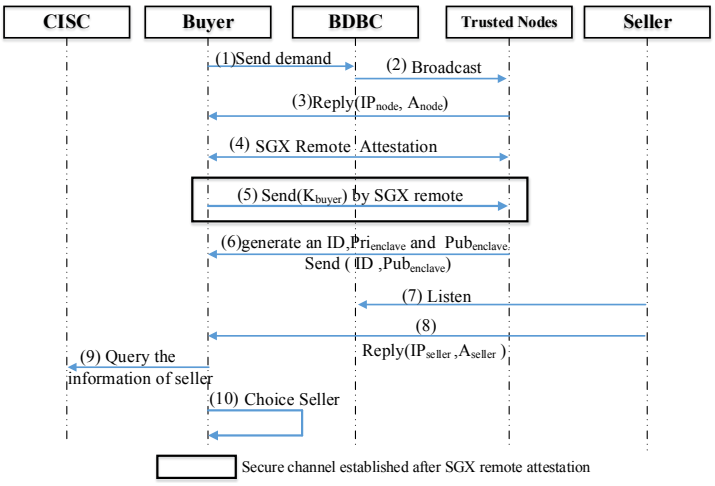

Fig. 5. The process of requirements matching in SDTP

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2928256, IEEE Transactions on Information Forensics and Security

- Step (6), ID, $\text{Pri}_{enclave}$ and $\text{Pub}_{enclave}$ are generated in Enclave. Then SGX seal method to store <ID, $\text{Pri}_{enclave}$, $\text{Pub}_{enclave}$>.
- Step (9), buyer will query seller's information by call CISC to judge the seller is good or bad. Not only this, The buyer can also directly query a specific transaction on the blockchain to further verify the authenticity of the information provided by CISC.
- The first n trusted nodes which have completed (3)-(6) will be selected by buyer to run $\text{C}_{analysis}$. It's worth noting that seller and trusted nodes may response to BDBC simultaneously. For convenience of expression, we display them separately in (3) and (8).
- Step (10),buyer will select seller based on the information(e.g. comment) in CISC. Malicious seller won't be selected by buyer.

E. Execution Preparation

Before executing $\text{C}_{analysis}$, trusted nodes need to do some preparations such as transmission of seller's decryption key and confirmation of the rewards that the trusted node and the seller can receive after the execution of the contract. The process of execution preparation is shown in Figure 6.

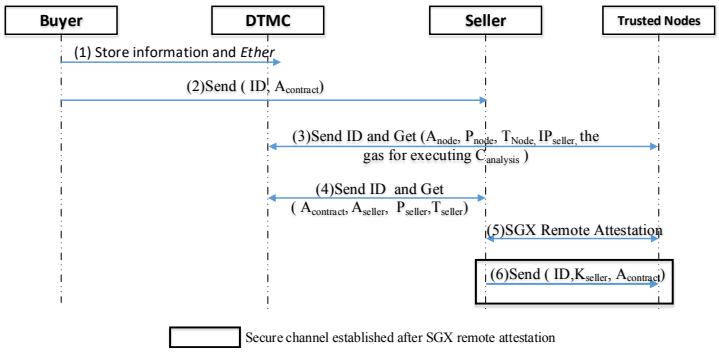

Fig. 6. The process of execution preparation in SDTP

- Step (1), buyer stores ID, $\text{A}_{contract}$, A_{node}, A_{seller}, P_{node}, P_{seller}, IP_{seller}, T_{node}, T_{seller}, $\text{Pub}_{enclave}$ into DTMC. Note that ID is the unique identifier of this execution contract. At the same time, pre-stored Ether will be divided into four parts: the gas for executing $\text{C}_{analysis}$, node reward, and seller reward. The reward will be transferred to the corresponding address.P_{node}, P_{seller}, T_{node}, T_{seller} are specify the unit price for processing unit data and the upper limit for rewards, which prevents the seller and trusted nodes from sending and processing additional useless data for additional rewards. $\text{Pub}_{enclave}$ is used to verify whether the information sent by the trusted node is generated within the Enclave to prevent the trusted node from forging the result information to obtain the reward. Finally IP_{seller} is used for the trusted nodes to accept a specific data seller, thereby preventing everyone from performing steps (5) and step (6).
- Step (3)(4), seller and trusted nodes will determine whether DTMC's response is in line with their expectations by querying the information of step(1).
- Step(5),The trusted nodes will only perform SGX remote authentication with IP_{seller}, The number of times which IP_{seller} performs step (5) is limited in a transaction, preventing the attacker or malicious seller from performing step (5) multiple times to bring DOS attack.
- Step (6), information will be sent into the Enclave of nodes through the secure channel which is established after SGX remote attestation. After receiving the information from seller, nodes will generate two maps in Enclave: <ID, $\text{A}_{contract}$, K_{seller}. Later the map is stored outside the Enclave via SGX's seal method.

F. Contract Execution

After execution preparation, trusted node now can execute $\text{C}_{analysis}$. The entire contract execution process is shown in Figure 7:

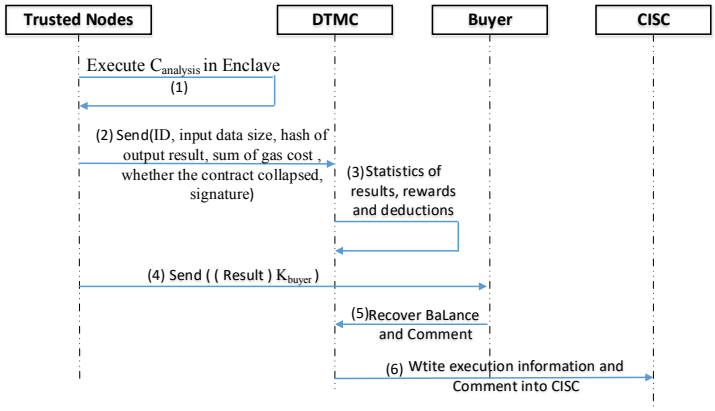

Fig. 7. The process of contract execution in SDTP

- Step (1), Before execute contract, enclave in trusted node will first use unseal method to get<ID, $\text{A}_{contract}$, K_{seller}>and then use ID to select corresponding $\text{A}_{contract}$ and K_{seller},then put contract into enclave according to $\text{A}_{contract}$. After that read the encrypted raw data of the seller into the Enclave and decrypt it with K_{seller}. Then EVM can use contract to process the seller's raw data. After the result is obtained, an additional charge will be made based on the gas consumed by the smart contract. The amount of the charge is related to the size of the output data. In the final signature process, Enclave uses $\text{Pri}_{enclave}$ to sign ID, input data size, hash of output result, sum of gas cost, whether the contract collapsed, generate the signature and then output ID, input data size, hash of output result, sum of gas cost. Those information will send to DTMC for consensus.
- Step (3), DTMC will perform signature verification to determine whether the sender belongs to the nodes selected by buyer. DTMC will count received data and the result with the most votes will be regarded as the final result. The node who votes the final result is regarded as an honest one and will be rewarded by buyer. Rewards for A_{node} and A_{seller} are calculated according to the size of the processing data combined with P_{node}, P_{seller}. Since malicious seller may send a large amount of redundant data to get more reward, DTMC will use T_{seller} and T_{node} as the upper bound of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2928256, IEEE Transactions on Information Forensics and Security

received data size when it calculates sellers' and trusted nodes' reward.Finally, the cost of gas consumed by EVM will be deducted.

- Step (5), When the buyer retrieves the balance, he must enter the evaluation of the smart contract and the data seller.
- Step (6), DTMC will write execution result (successful/fail), the buyer's comment, buyer's deposit Ether, the cost of run EVM, the input data size, the output data size and so on into CISC to help better evaluate seller. In order to encourage buyer to make an honest comment we can use existing research solutions which is mentioned in Section IV-B.This is not the focus of our research.CISC will use the existing research methods [30] [31] to select opinion spam.

VI. SECURITY ANALYSIS

In this section, we combine security analysis with Table I in Section III. We first analyze DOS attacks because all roles may launch this attack. They only need to constantly send transaction to call related smart contracts (e.g. BDBC, DTMC) to consume network resources. After that, we analyze the possible attack of every single malicious role, and then we move to the attack caused by the collusion of malicious roles. It is worth noting that when multiple roles collude, there will be some additional attacks in addition to the attack that a single role may initiate. When analyzing collusion attack, we only consider those additional attacks.

DOS Attack. On the one hand, an attacker (can be any role in SDTP) may consume SDTP's network resources by constructing multiple transactions to invoke BDBC. The attacker may broadcast his demand through BDBC continuously and do not perform any subsequent operations. To solve the problem, an additional fee will be charged by BDBC as the cost of broadcast. On the other hand, an attacker may consume network resources by depositing an extremely small amount of gas to DTMC because an exception may be thrawn due to insufficient gas during execution. To illustrate, communication such as key transmission needs to be performed before smart contracts are executed, and it's hard to charge a fee for communication immediately. Therefore, an attacker could use this to construct a DOS attack to waste network resource. SDTP defend this attack by a minimum pre-deposit to DTMC from buyer. We also impose restrictions on the seller's ability to transmit decrypted raw data keys. Only certain seller can perform remote authentication and the number of remote authentications performed in a transaction is limited.

Single Malicious Buyer. A malicious buyer can do two kind of things in SDTE. The first kind of thing is to write a malicious smart contract to steal or resell the original data or data analysis results. The second kind of thing is to deny the execution expense for seller and selected trusted nodes.

As for the malicious contract, single malicious buyer has two methods to do evil. The first is that smart contracts written by himself could call other smart contracts and leak seller's raw data. In order to solve this problem, we have made restrictions on EVM in Enclave and forbid contracts' ability to make calls. The second one is that smart contracts written by buyer may output the seller's raw data directly. To solve this problem, we made a charge for the output of smart contracts. If the buyer directly outputs the complete(or the part of) result, then the cost is huge first, and secondly, even if the buyer spends a huge amount of money to obtain the source data, then this is equivalent to buyer paying a huge cost to obtain the copyright of seller's complete (or the part of)raw data, which is acceptable for the seller. In the future, source data tracking in Enclave to further enhance the security of sellers raw data.

As for the buyers deny the execution expense, the buyer's expenses include three parts: selected trusted nodes reward, seller reward, and the gas for EVM to run smart contracts.

When the buyer cheats on the selected trusted nodes reward, (s)he may tamper with the address of the reward nodes in DTMC or sends only part of the selected trusted nodes' addresses to DTMC (e.g. buyer needs n nodes to execute a contract, but writes m nodes address to DTMC, m<n). Also, the buyer may modify the P_{node} stored in DTMC to reduce the reward rate for selected trusted nodes. In order to solve the problem in SDTP, the selected trusted nodes will query DTMC how much (s)he can get to perform the designated smart contract before executing it. Once his/her address is not in the reward list in DTMC or P_{node} doesn't meet his expectation, the selected trusted nodes won't execute the contract.

When the buyer cheats on the seller reward, he may modify A_{seller} and P_{seller} stored in DTMC. To protect the interests of himself, seller will query whether A_{seller} and P_{seller} is in line with his expectation before sending data to selected trusted nodes.

Because the DTMC deducts the expense of EVM execution based on the execution result, this process is completed by the smart contract, so the data buyer cannot deny the expense of the EVM.

Single Malicious Seller. In order to obtain more rewards, the seller may cheat on the source data. He may transmit redundant data to selected trusted nodes. For example, the buyer needs 100 units of data. In order to obtain additional rewards, the seller may send the 150 units of data to selected trusted node. To solve this problem, buyer could set an expected T_{seller}. In the above example, T_{seller} is precisely the maximum total price of the 100 units of data. Thus, seller will not get extra revenue even he sends extra data. In addition, the seller may also send data that does not match the buyer's needs to defraud profits. In this case, the buyer can obtain the seller's relevant comments from the CISC before purchasing the data. These comments are credible because we use existing research programs [30] [31] to encourage buyers who have previously traded with the seller to make an honest evaluation. In other word, buyer can select seller before choose the seller based on the information of CISC

Malicious Nodes. Malicious nodes do evil mainly for two purposes: obtain data and get more rewards. The key data obtained by a malicious node includes two aspects: data analysis result and the source data. These data are transmitted to selected trusted nodes in the form of ciphertext encrypted by keys (e.g. K_{buyer}, K_{seller}). Also, the encryption keys are passed into the Enclave through a secure channel established after remote authentication. What's more, the decryption and processing of these two kinds of data are performed within Enclave. Therefore, malicious selected nodes have no chance obtaining the plaintext of these data.

As for rewards, unselected nodes may copy the execution results (e.g. ID, input data size, hash of output result, sum of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2928256, IEEE Transactions on Information Forensics and Security

gas cost) of selected nodes. They could pretend to be selected nodes and tell DTMC execution information to receive rewards. In order to prevent such a situation, the buyer will store selected nodes' address into DTMC in advance. DTMC will judges that whether the sender of the information is the node specified by buyer.

What's more, malicious selected nodes may collaborate and report wrong execution result to DTMC together (e.g. the cost to run contract) to obtain more rewards. To solve this problem and ensure that the execution results that DTMC receives are not tampered with, $\text{Pri}_{enclave}$ is used to sign execution information in Enclave. $\text{Pri}_{enclave}$ is produced within the Enclave and stored using SGX seal technology, ensuring that $\text{Pri}_{enclave}$ is only known to Enclave itself. $\text{Pub}_{enclave}$ is used to verify the the signature produced with $\text{Pri}_{enclave}$. $\text{Pub}_{enclave}$ is passed to the buyer through the secure channel of the SGX remote attestation which ensures that the buyer can receive $\text{Pub}_{enclave}$ correctly. After that, buyer will store $\text{Pub}_{enclave}$ into DTMC for verifying signature from $\text{Pri}_{enclave}$ after DTMC receives the result information form selected nodes. In this way, nodes can't falsify execution results.

Collusion between the two Malicious Roles. When buyer and seller collude, In this case, they want to use the computing resources in SDTP, but do not want to pay to the selected trusted nodes and EVM to execute the smart contract. As mentioned above, the selected trusted nodes will call the DTMC to check before executing the smart contract, so even if the buyer and the seller collude, it is impossible to deny. They won't use malicious contract because they want to get the result.

When nodes and a buyer collude, what they can do is a malicious buyer can designate malicious selected nodes as the nodes to execute a smart contract. Seller's data will all run on malicious nodes. Before send data to selected nodes, the seller will only pass the key into the Enclave after the SGX remote authentication is passed. Nodes cannot steal and tamper with private data (see chapter6-Malicious Nodes).So the seller's raw data won't be leaked. In addition, seller will query DTMC whether he rewards is in line with his expectations before running a smart contract. Therefore, malicious node and buyer have no chance stealing and tampering with data and deny.

When nodes and a seller collude, the worst case is that all the nodes selected by buyer are malicious. In this case, they may want to steal buyer's analysis results , get extra rewards, send useless data. However, buyer sets the maximum value for the reward of seller and nodes. Enclave prohibits the call between contracts. Also, Enclave will sign the execution result information and ensure that the execution result will not be tampered with. Thus, buyer's interest could be guaranteed.

VII. IMPLEMENTATION AND EVALUATION

K-NearestNeighbor (KNN) [20], [34]algorithm is a very classic machine learning algorithm, so we implement KNN algorithm in the form of smart contract to measure the performance of SGX-protected EVM. The KNN algorithm we use is to calculate the type of input based on the type of the nearest three points to input points in a X-Y-Z space. Moreover, we have manually implemented a 256-bit integer operation library because SGX SDK [35] cannot use some library of the Original EVM(e.g. fenv.h).

In our system, there are four process in total: contract deployment, requirements matching, execution preparation and contract execution. We test the four process' time in total. we also test two main extra cost in execution preparation and contract execution. The first is SGX remote attestation which establishes a secure channel and transmits secret keys. The second is the execution cost for running a smart contract in a secure environment rather than in a traditional EVM.

We use two machines to simulate SDTE, one represents the the user (include seller, buyer, here we let $\textbf{Role}_{seller}$ and $\textbf{Role}_{buyer}$ denote them), the other one represents trusted nodes ($\textbf{Role}_{trust}$). $\textbf{Role}_{seller}$ and $\textbf{Role}_{buyer}$ are ordinary machines without SGX hardware environment and installs SGX SDK [35] to use some SDK libraries (e.g. libsgx_ukey_trading.a) to complete the remote attestation process. $\textbf{Role}_{seller}$ and $\textbf{Role}_{buyer}$ are equipped with 2.40GHz 64-bit Intel Xeon CPU E5-2630 v3 processor with 32-cores, 64 GB RAM and network interfaces with 1Gbps network speed. $\textbf{Role}_{trust}$ is Dell Precision Tower 3620 with Intel i7-6700U CPU and 32.00GB memory. The range of node's number we test in $\textbf{Role}_{trust}$ is from 3 to 19. We test and analyze two main extra items in the following parts.

A. Secure Key Transmission

We create **n** Enclaves in $\textbf{Role}_{trust}$ to represent the **n** nodes that execute data analysis contract in SDTP. Then we run SGX remote attestation & secure channel establishment & key transmission for 10000 times and get the average time, which is shown in Figure 8.

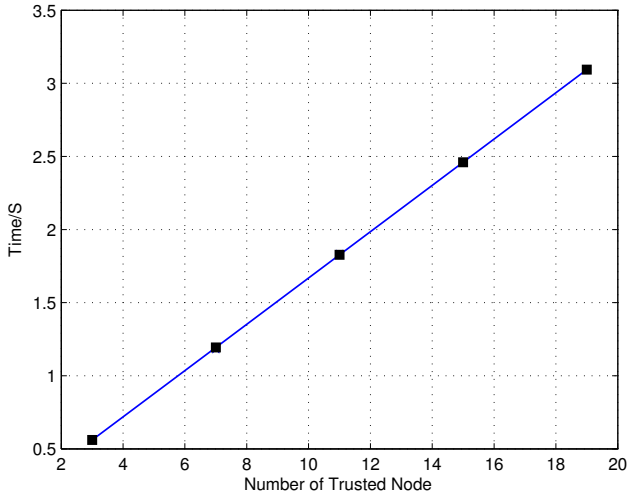

Fig. 8. The average time with the increase of $\textbf{Role}_{trust}$

Secure key transmission includes three parts, remote attestation, secure channel establishment and key transmission. With the number of $\textbf{Role}_{trust}$ increases, the total time increases linearly but the chance for trusted nodes collusion decreases at the same time. We believe that the time cost of passing the key is worthwhile, because only in this way can user ensure that the secret key cannot be acquired by malicious nodes in SDTP.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2928256, IEEE Transactions on Information Forensics and Security

B. Contract Execution in SGX

Figure 9 presents the overall time costs incurred in executing KNN on SGX and conventional EVM.

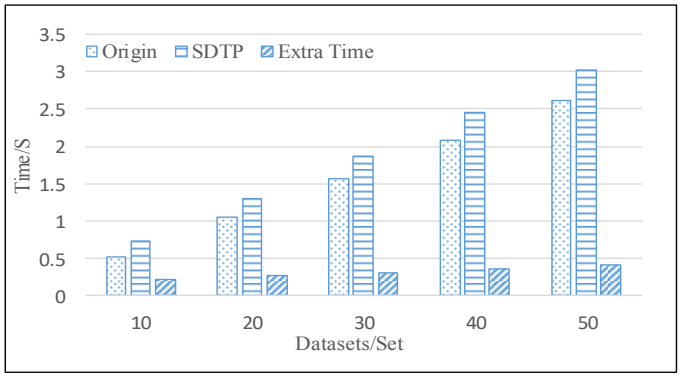

Fig. 9. Contract execution performance comparison between SDTP and conventional EVM

The total time required to run KNN in SGX comprises four key parts, namely: (1) reading of the smart contract bytecode into Enclave, (2) reading of buyer's data into Enclave and decrypting it, (3) running of the contract bytecode in EVM, and (4) encrypting of the processed results with the buyer's key. We calculate the average time overhead by executing the entire contract execution process on a Dell Precision Tower 3620 machine with different input data size for 10000 times. In (2) (4), we use the AES-256 symmetric encryption algorithm provided by wolfssl [36] to perform Enclave encryption and decryption. In (3), KNN processes 100 items of data at a time, where each item includes three 256-bit Integers. For convenience, we use "a set" to denote "100 items of data" and 10 sets means 1000 items of data (i.e., KNN has to run 10 times to handle such data).

Therefore, the additional time cost of contract execution in SGX increases linearly with the the number of data sets. Compared to conventional EVM, the additional overhead of SGX-protected EVM is spent in copying data within the Enclave from untrusted memory, decrypting source data and encrypting data analysis result within the Enclave. As the size of the input source data increases, the proportion of additional performance overhead becomes smaller because SGX-protected EVM requires more time to analyze the data by running the data analysis contract. When we use SGX-protected EVM to run more complex data analysis contracts, the time difference required by the SGX-protected EVM and conventional EVM decreases as the size of the input source data increases. In short, contract execution in SGX increases the security of execution significantly, without having an unrealistic overhead range.

C. Runtime

We build the Ethereum private chain on Ubuntu 16.04 to measure the total time required for the system, excluding the time spent by the role itself (for example, the decision time after the buyer obtains the seller's information from the CISC). Each data in the graph been tested 1000 times. To be as realistic as possible (in terms of the speed to generate new blocks in the public chain), we modify the difficulty in the genesis file to make the private chain produce a block in about 15 seconds. We select 10 trusted nodes to execute the data analysis contract and change the size of data sets to obtain different total time. In Steps (2) and (3) of the Contract Execution, 10 nodes will send 10 transactions to change the DTMC's state. We also assume that these 10 transactions are in the same block.

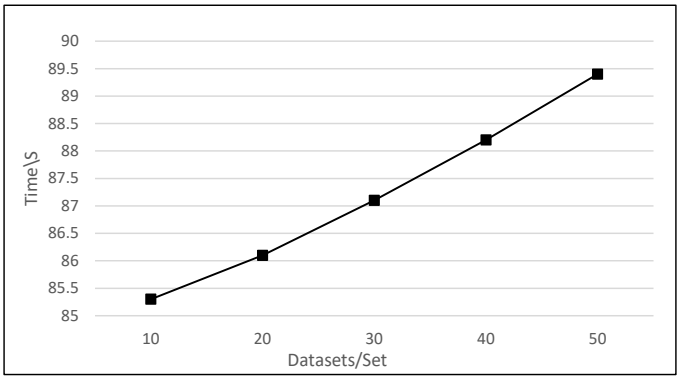

Fig. 10. Total runtime of SDTP

As shown in Figure 10, the total time increases linearly with the number of datasets. In SDTP, there are five steps (buyer deploys data analysis contract, buyer broadcasts demand by BDBC, etc) that change the state of contract, and at each time change the state costs SDTP approximately 15 seconds to reach a consensus. Combined with Figures 8 and 9, we determine that the main cost is due to calling contract (e.g. DTMC) through sending a transaction on the SDTP to change the state of a contract.

VIII. CONCLUSION

We presented SDTE, a blockchain-based ecosystem to complement and mitigate limitations in existing data trading market. Specifically, we introduced a paradigm shift where a buyer obtains the result of the data analysis rather than the actual dataset. We used blockchain to allow the tracing of unauthorized transactional modifications. We also built a SGX-based secure contract execution environment to protect the source data and the analysis result. We demonstrated the security of SDTE, including the capability to withstand attacks conducted by colluding arbitrary parties. SDTP was implemented based on Ethereum and SGX, and its performance evaluated.

Currently, SDTP is implemented using SGX SDK and C++ client, but Go client is also popular with Ethereum clients. Hence, future research includes having an implementation in Go client and extending the support to other environments (e.g., LLVM and JVM) in order to cater to a broader range of applications. Also as previously discussed, future work will include source data tracking in the enclave to further enhance the security of seller's raw data.

REFERENCES

[1] Y. Sun, H. Song, A. J. Jara, and R. Bie, "Internet of things and big data analytics for smart and connected communities," *IEEE Access*, vol. 4, pp. 766–773, 2016.

[2] X. Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, 2014.

[3] M. Mohammadi, A. I. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *arXiv preprint arXiv:1712.04301*, 2017.

[4] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: The next frontier for innovation, competition, and productivity," https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation, 2011.

[5] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, and D. Barton, "Big data: the management revolution," *Harvard business review*, vol. 90, no. 10, pp. 60–68, 2012.

[6] E. Dumbill, "Data markets compared-a look at data market offerings from four providers." http://radar.oreilly.com/2012/03/data-markets-survey.html, 2012.

[7] 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, no. 1, pp. 15 132–15 154, 2018.

[8] C. Zuo, J. Shao, J. K. Liu, G. Wei, and Y. Ling, "Fine-grained two-factor protection mechanism for data sharing in cloud storage," *IEEE Trans. Information Forensics and Security*, vol. 13, pp. 186–196, 2018.

[9] M. Felici, T. Koulouris, and S. Pearson, "Accountability for data governance in cloud ecosystems," in *Proceedings of the 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*. Los Alamitos, CA, USA: IEEE Computer Society, 2013, pp. 327–332.

[10] T. Jung, X. Li, W. Huang, J. Qian, L. Chen, J. Han, J. Hou, and C. Su, "Accounttrade: Accountable protocols for big data trading against dishonest consumers," in *Proceedings of the IEEE INFOCOM 2017 - IEEE Conference on Computer Communications (INFOCOM)*. Piscataway, NJ, USA: IEEE, 2017, pp. 1–9.

[11] K. Liang, W. Susilo, and J. K. Liu, "Privacy-preserving ciphertext multi-sharing control for big data storage," *IEEE Trans. Information Forensics and Security*, vol. 10, pp. 1578–1589, 2015.

[12] J. Chen and Y. Xue, "Bootstrapping a blockchain based ecosystem for big data exchange," in *Proceedings of 2017 IEEE International Congress on Big Data (BigData Congress)*. Los Alamitos, CA, USA: IEEE Computer Society, 2017, pp. 460–463.

[13] Federal Trade Commission, "Data brokers: A call for transparency and accountability," https://www.ftc.gov/system/files/documents/reports/data-brokers-call-transparency-accountability-report-federal-trade-commission-may-2014/140527databrokerreport.pdf, 2014.

[14] China Hangzhou Credit Data Technology Co., Ltd. , "Gxb blockchain white paper," https://ico.gxb.io/download/GXB_Blockchain_White_Paper_v1.2_EN.pdf, 2017.

[15] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. New York, NY, USA: ACM, 2016, pp. 270–282.

[16] J. Guarnizo and P. Szalachowski, "Pdfs: Practical data feed service for smart contracts," *arXiv preprint arXiv:1808.06641*, 2018.

[17] V. Buterin, "Schellingcoin: A minimal-trust universal data feed," https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/, 2014.

[18] 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.

[19] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Trans. Information Forensics and Security*, vol. 14, pp. 331–346, 2019.

[20] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 4, pp. 580–585, 1985.

[21] N. Szabo, "Smart contracts," http://szabo.best.vwh.net/smart.contracts.html, 1994.

[22] Buterin and Vitalik, "Ethereum: a next generation smart contract and decentralized application platform," http://ethereum.org/ethereum.html, 2017.

[23] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," in *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy(HASP)*. New York, NY, USA: ACM, 2013, pp. 10:1–10:1.

[24] I. Anati, S. Gueron, S. P. Johnson, and V. R. Scarlata, "Innovative technology for cpu based attestation and sealing," https://software.intel.com/en-us/articles/innovative-technology-for-cpu-based-attestation-and-sealing, 2013.

[25] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. Del Cuvillo, "Using innovative instructions to create trustworthy software solutions," in *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy(HASP)*. New York, NY, USA: ACM, 2013, pp. 11:1–11:1.

[26] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich, "Vc3: Trustworthy data analytics in the cloud using sgx," in *Proceedings of the 2015 IEEE Symposium on Security and Privacy(SP)*. San Jose, CA, USA: IEEE, 2015, pp. 38–54.

[27] N. Weichbrodt, A. Kurmus, P. Pietzuch, and R. Kapitza, "Asyncshock: Exploiting synchronisation bugs in intel sgx enclaves," in *Proceedings of the 21st European Symposium on Research in Computer Security(ESORICS)*. Cham, German: Springer Verlag, 2016, pp. 440–457.

[28] S. Lee, M.-W. Shih, P. Gera, T. Kim, H. Kim, and M. Peinado, "Inferring fine-grained control flow inside sgx enclaves with branch shadowing," in *Proceedings of the 26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC, Canada: USENIX Association, 2017, pp. 557–574.

[29] S. Shinde, Z. L. Chua, V. Narayanan, and P. Saxena, "Preventing page faults from telling your secrets," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security(ASIA CCS)*. New York, NY, USA: ACM, 2016, pp. 317–328.

[30] B. L. Nitin Jindal, "Opinion spam and analysis." in *Proceedings of the 1st ACM International Conference on Web Search and Data Mining(WSDM)*. Stanford, CA, USA: ACM, 2008, pp. 219–230.

[31] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, "Finding deceptive opinion spam by any stretch of the imagination." in *Proceedings of The 49th Annual Meeting of the Association for Computational Linguistics(ACL)*. Portland, Oregon, USA: The Association for Computer Linguistics, 2011, pp. 301–319.

[32] V. P. Kemerlis, G. Portokalidis, K. Jee, and A. D. Keromytis, "libdft: practical dynamic data flow tracking for commodity systems," in *Proceedings of the 8th International Conference on Virtual ExecutionEnvironments (VEE)*. London, UK: ACM, 2012, pp. 121–132.

[33] V. Pappas, V. P. Kemerlis, A. Zavou, M. Polychronakis, and A. D. Keromytis, "Cloudfence: Data flow tracking as a cloud service," in *Proceedings of Research in Attacks, Intrusions, and Defenses - 16th International Symposium(RAID)*. Rodney Bay, Saint Lucia: Springer, 2013, pp. 411–431.

[34] H. Wu, L. Wang, and T. Jiang, "Secure and efficient *k*-nearest neighbor query for location-based services in outsourced environments," *SCIENCE CHINA Information Sciences*, vol. 61, p. 3, 2018.

[35] Intel Corporation, "Intel® software guard extensions sdk for linux os*," https://download.01.org/intel-sgx/linux-2.0/docs/Intel_SGX_SDK_Developer_Reference_Linux_2.0_Open_Source.pdf, 2017.

[36] wolfSSL Inc., "Wolfssl-embedded ssl. library for applications, devices, iot, and the cloud," https://www.wolfssl.com/, 2018.